Theory of Hybrid Systems Informatik 2
RWTH AACHEN UNIVERSITY

The present work was submitted to the LuFG Theory of Hybrid Systems

BACHELOR OF SCIENCE THESIS

EMBEDDING THE VIRTUAL SUBSTITUTION IN THE MCSAT FRAMEWORK

Jasper Nalbach

Examiners:
Prof. Dr. Erika Ábrahám
Prof. Dr. Jürgen Giesl

Aachen, 11th August, 2017

Abstract

Satisfiability modulo theories (SMT) is a technology for checking the satisfiability of logical formulas over some theories. Recently the novel framework *model-constructing satisfiability calculus (mcSAT)* was introduced showing promising results for solving formulas from the theory of *non-linear arithmetic* in combination with the *cylindrical algebraic decomposition (CAD)* method. This thesis proposes an embedding of the incomplete *virtual substitution (VS)* method for checking the satisfiability of quantifier-free non-linear arithmetic formulas in the mcSAT framework.

Eidesstattliche Versicherung

Name, Vorname

Matrikelnummer (freiwillige Angabe)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel

selbständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Ort, Datum

Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Ort, Datum

Unterschrift

Acknowledgements

I am grateful for the opportunity to work on this fruitful topic and for the insights given to me into scientific work. Special thanks to Erika Ábrahám for supervising this thesis and for her time to discuss arising issues. Also thanks to the additional examiner Jürgen Giesl.

Thanks to my family for encouraging and supporting me.

Contents

Chapter 1

Introduction

Despite the hardness of the propositional satisfiability (SAT) problem, there has been intensive research over the past decades resulting in reasonably fast SAT solvers. The most widespread class are *conflict-driven clause-learning (CDCL)* solvers, which build on the *DPLL* [DP60, DLL62] framework. Furthermore, the problem statement has been extended for first-order logic, namely *satisfiability modulo theories (SMT)* solving. These formulas are in most cases quantifier-free, e.g. they are Boolean combinations of constraints from some theories.

A successful technique for checking such formulas for satisfiability is the *DPLL(T)* framework, which is a modular procedure switching between a SAT solver and one or more theory solvers. The SAT solver is responsible for the satisfaction of the Boolean structure of the formula whereas the theory solver is consulted to assure satisfaction also in the underlying theory. In this framework, the theory solver checks a conjunction of theory constraints for consistency. In case of an inconsistent input, it generates a theory lemma explaining the inconsistency.

Various methods for solving non-linear arithmetic have been implemented as theory solvers for DPLL(T), namely the *cylindrical algebraic decomposition (CAD)* [Col75] and incomplete methods such as *interval constraint propagation* [GGI+10, HR97] or the *virtual substitution (VS)* method [Wei97, CA11, Cor16].

Lately, the *model-constructing satisfiability calculus (mcSAT)* [DMJ13] has been introduced as an alternative approach to DPLL(T). Various decision procedures have been implemented for mcSAT. For quantifier-free linear real arithmetic, the Fourier-Motzkin variable elimination has been implemented [DMJ13], and the implementation of the CAD method for solving quantifier-free non-linear real arithmetic shows promising results [JdM12].

In the following we embed the virtual substitution into the mcSAT framework for solving *quantifier-free non-linear real arithmetic*. In Chapter 2, a short introduction to the mcSAT framework and the virtual substitution method is given. In Chapter 3, we present our embedding of the virtual substitution method into the mcSAT framework. Then, we compare our approach with the CAD in Chapter 4, and give some ideas for further improvements in Chapter 5.

Chapter 2

Preliminaries

2.1 Non-linear real arithmetic

In the following let $\mathbb{R}$ denote the set of all real numbers and let $\mathbb{N}_0$ denote the set of all natural numbers including zero. We define the syntax of quantifier-free non-linear real arithmetic formulas as follows:

$\langle formula \rangle ::= \langle constraint \rangle \mid \langle formula \rangle \wedge \langle formula \rangle \mid \neg \langle formula \rangle$

$\langle constraint \rangle ::= \langle term \rangle \ \langle op \rangle \ \langle term \rangle$

$\langle op \rangle ::= < \mid =$

$\langle term \rangle ::= x \mid c \mid \langle term \rangle + \langle term \rangle \mid \langle term \rangle - \langle term \rangle \mid \langle term \rangle * \langle term \rangle$

where $x \in \mathit{Vars}$ is a real-valued variable and $c \in \mathbb{R}$. We use syntactic sugar such as the operators $>, \leq, \geq, \neq$ and logical connectives such as $\vee, \rightarrow$. The semantics is defined as usual.

The terms are called *polynomials*. The set of all polynomials with coefficients from $\mathbb{R}$ containing variables $x_1, \ldots, x_n \in \mathit{Vars}$ is denoted as $\mathbb{R}[x_1, \ldots, x_n]$. A polynomial p is called *univariate* if $p \in \mathbb{R}[x_1]$ for some variable $x_1 \in \mathit{Vars}$ and is called *multivariate* if $p \in \mathbb{R}[x_1, \ldots, x_n]$ with $n > 1$. A power product $\prod_{i=1}^{n}(x_i^{e_i})$ where $e_i \in \mathbb{N}_0$ is called a *monomial*. The *degree* of the variable x_i in the monomial $M = \prod_{i=1}^{n}(x_i^{e_i}), e_i \in \mathbb{N}_0$ is defined as $\deg_{x_i}(M) = e_i$ whereas the *total degree* of the monomial M is defined as $\deg(M) = \sum_{i=0}^{n} e_i$. The *normal form* of polynomial $p \in \mathbb{R}[x_1, \ldots, x_n]$ is defined as the term $\sum_{k=0}^{m} a_k * M_k = p$ for some $a_k \in \mathbb{R}$ and monomials M_k in variables $x_1, \ldots, x_n$. The *degree* of a variable x_i in p is defined as $\deg_{x_i}(p) = \max\{\deg_{x_i}(M_k) \mid k = 1, \ldots, m\}$. The *total degree* of p is defined as $\deg(p) = \max\{\deg(M_k) \mid k = 1, \ldots, m\}$. A variable x_{x_i} occurs *quadratically* in a polynomial p if $\deg_{x_i}(p) = 2$. A polynomial p is called *linear*, if $\deg(p) \leq 1$, otherwise it is called *non-linear*.

We define the sign of a real value as the function $\mathit{sgn} : \mathbb{R} \rightarrow \{-1, 0, 1\}$ where $\mathit{sgn}(d) = -1$ if $d < 0$, $\mathit{sgn}(d) = 0$ if $d = 0$ and $\mathit{sgn}(d) = 1$ if $d > 0$.

We can rewrite each constraint $p_1 \sim p_2$ to an equivalent constraint of the form $p \sim 0$ where $p, p_1, p_2 \in \mathbb{R}[x_1, \ldots, x_n]$ are multivariate polynomials and the relation symbol is $\sim \in \{<, >, \leq, \geq, =, \neq\}$. Also, we can reformulate a formula to not contain negations without introducing new atoms by using De Morgan's law and replacing relation symbols by their respective counterparts.

We denote the set of all polynomials occurring in a formula φ with $\mathit{Pols}(\varphi)$, the set of all constraints in φ with $\mathit{Constraints}(\varphi)$ and the set of all variables in φ with $\mathit{Vars}(\varphi)$.

An *assignment* for a set Vars of real-valued variables is a possibly partial function $\alpha : \mathit{Vars} \rightharpoonup \mathbb{R}$ where $\rightharpoonup$ denotes a partial function. If $\mathrm{dom}(\alpha) = \mathit{Vars}$ then α is called a *full assignment*, otherwise

it is called a *partial assignment*. An assignment $\alpha' : \mathit{Vars} \rightharpoonup \mathbb{R}$ is called an *extension* of $\alpha : \mathit{Vars} \rightharpoonup \mathbb{R}$ if $\mathrm{dom}(\alpha) \subset \mathrm{dom}(\alpha')$ and $\alpha'(x) = \alpha(x)$ for all $x \in \mathrm{dom}(\alpha)$. We denote the set of all possible assignments for a formula φ as $\mathit{Assigns}(\varphi)$. Let $\alpha : \mathit{Vars} \rightharpoonup \mathbb{R}$ be an assignment and $V \subseteq \mathrm{dom}(\alpha)$ be a set of variables, then the *restriction* of α to V is the assignment $\alpha \downarrow_V : V \to \mathbb{R}$.

Given a term t or a formula φ, we define its *interpretation* under an assignment α as $[\![t]\!]^\alpha$ respectively $[\![\varphi]\!]^\alpha$ as usual. For formulas we define $\alpha \models \varphi$ as $[\![\varphi]\!]^\alpha \equiv \mathit{true}$, that is φ is *satisfied* under α. If $[\![\varphi]\!]^\alpha \equiv \mathit{false}$, we say φ is *conflicting* under α or *inconsistent* with α. If a satisfying assignment for φ exists, we say φ is *satisfiable*, otherwise it is *unsatisfiable*. We will use the standard substitution $\varphi[t/x]$ replacing all free occurrences of x in φ by t. (Note that all occurrences are free as we do not allow quantifiers.)

Please note that in the following, we use $\models$ to denote semantic consequence and $\equiv$ to denote logical equivalence of formulas. The notations $\Longrightarrow$ and $\Longleftrightarrow$ are used in the mathematical sense for derivation, whereas $\to$ is used as logical connective in formulas.

2.2 Model-constructing satisfiability calculus (mcSAT)

We give a brief introduction to the novel technique called *model-constructing satisfiability calculus (mcSAT)*. For more details, we refer to [DMJ13].

The mcSAT framework searches for satisfying solutions for the input formula. While the satisfaction of the Boolean structure is assured in a similar way as in DPLL, in contrast to the standard SMT approach, an assignment for the theory variables is constructed simultaneously by "guessing" values for them. In case of a conflicting assignment, the theory solver needs to explain the conflict of a set of constraints under a given partial assignment to the theory variables. Comparing to the DPLL(T) framework, the input is not only a set of constraints whose consistency is to be checked but also an additional partial assignment. More precisely, the input to the theory solver is a set of theory constrains in variables $x_1, \ldots, x_n, y$ and an assignment α of values to $x_1, \ldots, x_n$ so that there exists no extension of α for y that satisfies all input constraints.

Explanations for theory conflicts Assume that the procedure has already assigned the variables $x_1, \ldots, x_n \in \mathit{Vars}$ to real values from $\mathbb{R}$. Let the assignment be denoted as $\alpha : \{x_1, \ldots, x_n\} \to \mathbb{R}$. Now assume the procedure cannot find a theory value for a variable y that is consistent with a set D of constraints. Formally we have

$$\alpha \not\models \exists y. \bigwedge_{l \in D} l$$

Then D is called a set of *conflicting constraints* under the assignment α. We define

$$A \equiv \bigwedge_{l \in D} l$$

as the conjunction of the conflicting constraints.

D is called *minimal* if no element can be removed from D without breaking the conflict, thus each constraint is part of the reason for the conflict. Formally, D is minimal if

$$\alpha \models \exists y. \bigwedge_{l \in D \setminus \{c\}} l \text{ for all } c \in D$$

Each explanation T needs to be valid and needs to exclude the current assignment, i.e. it must hold that $A \wedge T \to \bigvee_{i=1,\ldots,n} x_i \neq \alpha(x_i)$. Furthermore, mcSAT requires all constraints in

an explanation to be from a finite set called *finite basis* $\mathbb{B}$. Otherwise, the termination of the procedure is not guaranteed.

Definition 2.1 (Explanation). *Assume a (possibly minimal) set of conflicting constraints D containing variables $x_1,\ldots,x_n,y$ and an assignment $\alpha : x_1,\ldots,x_n \to \mathbb{R}$ so that for each extension α' of α for y, α' is conflicting with $A \equiv \bigwedge_{l\in D} l$. Then explain$(A,\alpha)$ is a theory lemma T so that*

- $\models T$,
- *T is of the form $A \to \varphi$ for some formula φ where $\alpha \models \neg\varphi$ and*
- *$l \in \mathbb{B}$ for all constraints l in T.*

Note that there are explanation functions that do not require a minimal set of conflicting constraints.

2.3 Virtual substitution

Virtual substitution (VS) [Wei97] is a quantifier-elimination procedure for linear and non-linear arithmetic formulas. It can be applied to arbitrary formulas with existential and universal quantifiers, but originally can only eliminate variables that appear at most quadratically in the formula. During the elimination of a variable, the degree of some other variables may be increased so that even those variables which appear at most quadratically in the input formula cannot be always eliminated. Despite this restriction, the VS method works efficiently for a number of examples.

2.3.1 Construction of test candidates

Partition into sign-invariant regions

A region in the Euclidean space is defined as follows:

Definition 2.2 (Region). *A* region *is a subset $A \subseteq \mathbb{R}^n$ of $\mathbb{R}^n$ for some $n \in \mathbb{N}$ that is open, connected and non-empty.*

Given a polynomial p and *Assigns*(p) as the set of possible assignments of variables in p, we partition the solution space into regions as described in the following. Thereby we assume a total order on the theory variables $x_1,\ldots,x_n$ and view assignments also as values $(\alpha(x_1),\ldots,\alpha(x_n))$ in $\mathbb{R}^n$.

Definition 2.3 (Sign-invariant region for a polynomial). *A region $A \subseteq \mathbb{R}^n$ is called* sign-invariant for a polynomial $p \in \mathbb{R}[x_1,\ldots,x_n]$, *if for all $\alpha_1,\alpha_2 \in A$ it holds that $sgn([\![p]\!]^{\alpha_1}) = sgn([\![p]\!]^{\alpha_2})$.*

In a sign-invariant region of a polynomial, the polynomial is either greater than, equal to or smaller than zero for all values in the region. Thus, the value of $p \sim 0$ where $\sim \in \{<,>,\leq,\geq,=,\neq\}$ does not change in a sign invariant region regardless of the relation $\sim$.

This can be extended to a set of polynomials: Let *Pols*(φ) be the set of all polynomials in the formula φ, we can intersect the sign invariant regions of all $p \in$ *Pols*(φ):

Definition 2.4 (Sign-invariant region for a formula). *A region $A \subseteq \mathbb{R}^n$ is called* sign-invariant for a formula φ *containing variables $x_1,\ldots,x_n$, if A is sign-invariant for all $p \in$ Pols(φ).*

Also, the value of φ does not change in a sign-invariant region for φ. As a consequence, it is sufficient to check one point from each sign-invariant region of φ to check the satisfiability of φ. As we can decompose the Euclidean space into a finite set of sign-invariant regions for each quantifier-free real arithmetic formula, this leads to the main idea of the virtual substitution: To eliminate a single variable, we generate a finite set of sample points (*test candidates*) and plug them into the formula (*substitution*).

Sample points for univariate formulas

First, we concentrate on the case of formulas where a single variable x occurs that we want to eliminate.

Given a polynomial $p \neq 0$ (the other case is trivial) in such a formula, recall that in a sign-invariant region for p, the polynomial p is either invariantly positive, zero or negative. This means, that the sign-invariant regions of p are the zeros $\xi_0, \ldots, \xi_n$ of p and the intervals between these zeros and on the left (right) of the smallest (largest) zero, in general:

$$(-\infty, \xi_0), [\xi_0, \xi_0], (\xi_0, \xi_1), [\xi_1, \xi_1], \ldots, [\xi_n, \xi_n], (\xi_n, \infty)$$

Since we only want to consider one sample point per region, we represent each sign-invariant region by its leftmost point. For closed intervals, this is straight forward. For left-open intervals, we either take a sufficiently small value $-\infty$ or a value infinitesimally close to the left bound d represented as $d + \epsilon$ where $\epsilon > 0$ is an infinitesimal value. For the polynomial p we get thus the sample points:

$$-\infty, \xi_0, \xi_0 + \epsilon, \xi_1, \ldots, \xi_n, \xi_n + \epsilon$$

For the open intervals, we could also search for suitable values to avoid the use of $-\infty$ and ϵ. However, this is too expensive and not possible for the multivariate case (which we will see later).

When looking at a formula φ, we proceed similarly as above but based on the zeros of *all* polynomials $p \in \mathit{Pols}(\varphi)$. I.e. we collect as sample points $-\infty$ and the values ξ and $\xi + \varepsilon$ for any zero ξ of any polynomial $p \in \mathit{Pols}(\varphi)$. Intuitively, the resulting sample points for φ are the points where a polynomial constraint $p \sim 0 \in \mathit{Constraints}(\varphi)$ changes its truth value.

Because the sample points cover all sign-invariant regions for φ, it holds that

$$\exists x.\varphi \equiv \bigvee_{r \in \{-\infty, \xi_0, \xi_0+\epsilon, \xi_1, \ldots, \xi_n, \xi_n+\epsilon\}} \varphi[r/x]$$

Sample points for multivariate formulas

The idea from the univariate case can be generalized to multivariate polynomials in n variables to obtain an equisatisfiable formula in $n-1$ variables. Given a variable x we want to eliminate, we construct a set of *test candidates* similar to the sample points from the univariate case.

The main difference between the sample points from above and the test candidates is that the zeros of the polynomials may be symbolic and depend on the remaining variables. Consequently, the order of the zeros is unknown and these zeros may only exist under certain *side conditions*.

We circumvent this by making use of $-\infty$ and ϵ instead of concrete values as representatives for the tested intervals. Moreover, we encode the given side conditions and the given sample points using equivalent symbolic expressions.

We define symbolic expressions for zeros of a formula in a variable x only for formulas that are quadratic in x, based on the solution equation for quadratic polynomial equations. Moreover,

no solution equations for zeros in cases with a degree higher than 4 are known. This causes the restriction of this procedure, though extensions of VS to arbitrary polynomial degrees exist.

There may be doubts if we still cover all relevant sign-invariant regions to prove satisfiability since the existence and ordering of the zeros is not known. However, if a zero does not exist, the corresponding sign-invariant regions do not exist. Moreover, the leftmost points of the resulting intervals of the intersection are the same no matter in which order they occur. Thus, the argumentation from the univariate case is still valid for the multivariate case.

The symbolic zeros are expressed as *square root expressions* as follows:

Definition 2.5 (Square root expression)**.** *A* square root expression *has the form*

$$\frac{p + q\sqrt{r}}{s}$$

where p,q,r,s are polynomials.

The zeros of a polynomial are given as:

Definition 2.6 (Zeros of a polynomial and their side conditions)**.** *Given a quadratic equation* $p = p_1x^2 + p_2x + p_3 = 0$ *where* p_1,p_2,p_3 *are polynomials not containing* x*, if* p *depends on* x*, i.e., if* $p_1 \neq 0 \vee p_2 \neq 0$*, then the solutions of* $p = 0$ *for* x *are:*

$$\begin{aligned}
x_0 &= -\frac{p_3}{p_2}, && \text{if } p_1 = 0 \wedge p_2 \neq 0 \\
x_1 &= \frac{-p_2 + \sqrt{p_2^2 - 4p_1p_3}}{2p_1}, && \text{if } p_1 \neq 0 \wedge p_2^2 - 4p_1p_3 \geq 0 \\
x_2 &= \frac{-p_2 - \sqrt{p_2^2 - 4p_1p_3}}{2p_1}, && \text{if } p_1 \neq 0 \wedge p_2^2 - 4p_1p_3 \geq 0
\end{aligned}$$

The set of symbolic zeros of p *is defined as*

$$\mathit{Zeros}(x,p) = \{x_0, x_1, x_2\}$$

The side condition of a zero $\xi \in \mathit{Zeros}(x,p)$ *is defined as*

$$sc(\xi) = \begin{cases} p_1 = 0 \wedge p_2 \neq 0 & \text{if } \xi = x_0 \\ p_1 \neq 0 \wedge p_2^2 - 4p_1p_3 \geq 0 & \text{if } \xi = x_1 \text{ or } \xi = x_2 \end{cases}$$

If $p_1 = 0$ and $p_2 = 0$, then any real value yields the same sign for p, thus we can take a single arbitrary value for x to cover this case.

As one can see, all zeros are square root expressions. This allows the definition of the set of representatives of all sign-invariant regions using square root expressions, ϵ and $-\infty$:

Definition 2.7 (Representatives and their side conditions)**.** *The* set of representatives for a variable x occurring at most quadratically in a constraint c *of the form* $p \sim 0$ *where* $\sim \in \{<, >, \leq, \geq, =, \neq\}$ *is defined as*

$$rs(x,c) = \{-\infty\} \cup \mathit{Zeros}(x,p) \cup \{\xi + \varepsilon \mid \xi \in \mathit{Zeros}(x,p)\}$$

The set of representatives of all sign-invariant regions for a variable x occurring at most quadratically in a formula φ *is defined as*

$$rs(x,\varphi) = \bigcup_{c \in Constraints(\varphi)} rs(x,c)$$

The side condition of a representative r *is defined as*

$$sc(r) = \begin{cases} sc(r') & \text{if } r = r' + \epsilon \\ sc(r) & \text{if } r \text{ is a zero} \\ true & \text{if } r = -\infty \end{cases}$$

where r' *does not contain* ε.

At this point, we can make an observation: Since we are only interested in satisfying assignments for a given constraint $p \sim 0$, we only need to consider sign-invariant regions that are solutions of a constraint. This means that based on the relation symbol $\sim$ in $p \sim 0$, some test candidates representing sign-invariant regions that cannot contain a solution can be excluded. Thus it is sufficient to choose the test candidates from $rs(x, p \sim 0)$ as follows:

- For $\sim \in \{<, >, \neq\}$, we can ignore all zeros and take only the open intervals $-\infty, \xi_0+\epsilon, \ldots, \xi_n+\epsilon$.
- For $\sim \in \{=\}$, we can ignore the intervals between the zeros and take only the zeros $\xi_0, \ldots, \xi_n$ and $-\infty$ for the case that all values are solutions.
- For $\sim \in \{\leq, \geq\}$, we know that if the constraint $p \sim 0$ is satisfied between two zeros, then it is also satisfied at a zero. Thus we take only the zeros $\xi_0, \ldots, \xi_n$ and $-\infty$.

We obtain a general definition for the set of test candidates:

Definition 2.8 (Test candidates)**.** *The* set of test candidates for a variable x in a constraint *of the form* $p \sim 0$ *where* $\sim \in \{<, >, \leq, \geq, =, \neq\}$ *is defined as*

$$tcs(x, p \sim 0) = \begin{cases} \{-\infty\} \cup Zeros(x, p) & \sim \in \{\leq, \geq, =\} \\ \{-\infty\} \cup \{\xi + \varepsilon \mid \xi \in Zeros(x,p)\} & \sim \in \{<, >, \neq\} \end{cases}$$

The set of test candidates for a variable x occurring at most quadratically in a formula φ *is defined as*

$$tcs(x, \varphi) = \bigcup_{c \in Constraints(\varphi)} tcs(x,c)$$

We formulate our observation which is proven in [Wei97]. In the following theorem, we make use of the usual semantics for division, square root and non-standard reals including infinitesimals and infinity:

Theorem 2.1. *Given some* $r \in rs(x,\varphi) \backslash tcs(x,\varphi)$ *and an* $\alpha : Vars(\varphi) \backslash \{x\} \to \mathbb{R}$ *such that* $\varphi[\alpha(r)/x]$ *is satisfiable, then there exists some* $t \in tcs(x,\varphi)$ *such that* $\varphi[\alpha(t)/x]$ *is satisfiable.*

Thus, given a formula φ containing a variable x at most quadratically is satisfiable (valid) if and only if for one (each) test candidate $t \in tcs(x, \varphi)$ its side condition $sc(t)$ holds and given $x = t$, φ is satisfiable.

2.3.2 Virtual substitution

To eliminate a variable x in a formula φ, all occurrences of x in φ have to be substituted by its test candidates. However, the generated test candidates contain expressions that are not part of our theory and thus, naive replacement would lead to a formula which cannot be further evaluated.

Let x be a variable that occurs at most quadratically in a constraint c and given a test candidate t, virtual substitution defines rules to create an equisatisfiable formula $\varphi' \wedge sc(t)$ to $c \wedge "x = t"$ which does not contain x and does not add any new variables. Such a formula φ' is denoted as

$$c[t//x]$$

We define $\varphi[t//x]$ for a formula φ analogously, where we replace each constraint c in φ by $c[t//x]$.

The choice of the rule used to compute $c[t//x]$ depends on the relation symbol used in c and the form of t:

- $-\infty$
- contains ϵ
- square root expression $\frac{p+q\sqrt{r}}{s}$ with $r = 1$ (fractions)
- square root expression $\frac{p+q\sqrt{r}}{s}$ with $r \neq 1$.

The substitution rules are given in [Wei97] and fulfill the following requirement:

Theorem 2.2 (Virtual substitution). *Let φ be a formula, $x \in Vars(\varphi)$ appears at most quadratically in φ and $t \in rs(x,\varphi)$. Then it is valid that $Vars(\varphi[t//x] \wedge sc(t)) \subseteq Vars(\varphi) \setminus \{x\}$ and*

$$\alpha \models \varphi[t//x] \wedge sc(t) \iff \alpha \models \varphi[\alpha(t)/x] \wedge sc(t)$$

for all $\alpha : Vars(\varphi) \to \mathbb{R}$.

2.3.3 Quantifier elimination procedure

To eliminate a variable, we use the following theorem:

Theorem 2.3 (Variable elimination). *Let φ be a quantifier-free formula in which x occurs at most quadratically, then:*

$$\exists x.\varphi \equiv \bigvee_{t \in tcs(x,\varphi)} (\varphi[t//x] \wedge sc(t))$$

Universally quantified variables can be eliminated similarly.

The complexity of the resulting formula can be estimated as follows:

Theorem 2.4 (Complexity). *Let $deg(p)$ denote total degree of a polynomial p and for a formula φ let $D(\varphi) = max(1, max\{deg(p) \mid p \in Pols(\varphi)\})$. Let $at(\varphi)$ denote the number of constraints in φ.*

Assume a formula φ and a variable x which occurs at most quadratically in φ. Let φ' be the formula that results from the application of Theorem 2.3 to eliminate x in φ. Then

$$\begin{aligned} D(\varphi') &\leq 6D(\varphi) - 8 && \text{for } D(\varphi) \geq 2 \\ at(\varphi') &\leq 16at(\varphi) + 63at(\varphi)^2 && \text{for } D(\varphi) = 1 \end{aligned}$$

If x occurs only linearly in φ, the bounds can be improved:

$$\begin{aligned} D(\varphi') &\leq 2D(\varphi) - 1 \\ at(\varphi') &\leq 7at(\varphi) + 6at(\varphi)^2 \end{aligned}$$

By repeatedly applying Theorem 2.3, all variables that fulfill the requirement can be eliminated. Note that due to Theorem 2.4, when eliminating a variable, the degree of the remaining variables may be increased. Thus, the applicability of the procedure to an example depends on the structure of the formula, among others on the ordering of the eliminated variables.

Given the number of added constraints in Theorem 2.4, the overall complexity of eliminating multiple variables is bounded from above by a single exponential bound in the number of quantifiers.

2.3.4 Search tree

We visualize the virtual substitution using a search tree:

Definition 2.9 (Virtual substitution tree)**.** *Given a quantifier-free formula φ containing variables $x_1, \ldots, x_n$ and w.l.o.g. we assume the elimination order $x_1 < x_2 < \cdots < x_n$, the* virtual substitution tree *$T(\varphi, x_1, \ldots, x_n)$ is generated as follows:*

- *The root node contains the input formula φ.*
- *If $\varphi \notin \{true, false\}$, for each test candidate $t \in tcs(x_1, \varphi)$ a child node is generated. Each child node for a test candidate t contains the subtree $T(\varphi[t//x_1] \wedge sc(t), x_2, \ldots, x_n)$.*

A formula φ with variables $x_1, \ldots, x_n$ is satisfiable if and only if a leaf node of $T(\varphi, x_1, \ldots, x_n)$ contains the formula *true*.

Example 2.1. *Let φ be defined as*

$$\varphi \equiv \underbrace{(x-2)^2 + (y-2)^2 - 1 < 0}_{c_1} \wedge \underbrace{x - y = 0}_{c_2}$$

The gray line in Figure 2.1 denotes the solutions for this problem, whereas the virtual substitution tree is shown in Figure 2.2.

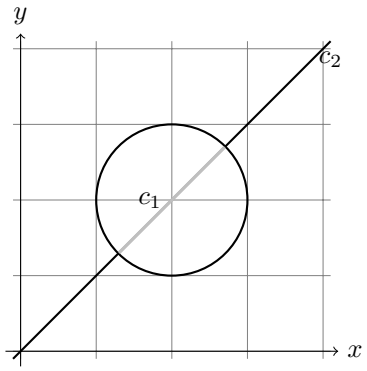

Figure 2.1: Satisfiable regions

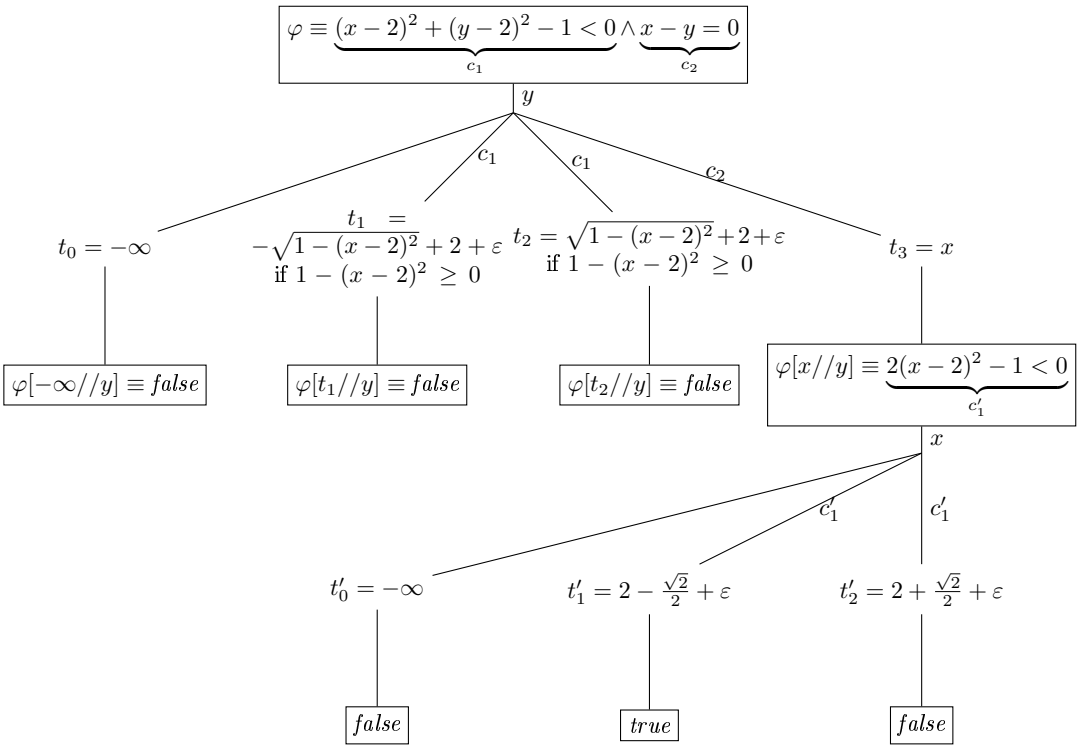

Figure 2.2: Virtual substitution tree

Chapter 3

Using virtual substitution for explanations for mcSAT

To integrate the virtual substitution method into the mcSAT framework, we have to provide an explanation function according to Definition 2.1. Given the conjunction of the conflicting constraints A and the current (partial) assignment α as input, it is known that $\exists y.A$ is conflicting with α.

The first step is to eliminate the only unassigned variable y in A using Theorem 2.3. At this point, the lemma

$$A \to \bigvee_{t \in tcs(y,A)} (A[t//y] \wedge sc(t))$$

could already be returned as an explanation lemma, since it is a tautology based on the correctness of the virtual substitution method. However, this does not only explain the *current* conflict, but is a general condition for the existence of a value for y satisfying A. In theory, a general explanation excludes bigger regions, but in practice, it may be desirable to generate an explanation that is specific to the current assignment in order to obtain a formula that is probably shorter and computationally easier to be used by the framework, e.g. for assigning values to theory variables.

Instead of eliminating the assigned variables $x_1, \ldots, x_n$ from A by substituting all test candidates for each variable, which could also be used to generate an explanation but too expensive, we aim for the following: We first use the VS to eliminate y from the input formula, which gives us a tree consisting of a root node (representing the input formula) and k children of the root (the ith child node representing $\varphi[t_i//y]$ where t_i is the ith test candidate for y). For each of these child nodes, instead of computing the complete VS tree below it, we compute just a single (full or partial) path rooted in it that represents the current assignment α. Finally, we define a logical description of the paths in this partial VS tree to replace the right-hand side of the implication above by a formula that is more specific to α. A schematic illustration of such a tree is given in Figure 3.1. This idea has something in common with the generation of explanations via the CAD method as proposed in [JdM12], as it also describes a sign-invariant region where the current assignment lays in.

The procedure can be sketched as follows: Because y is not in the domain $\mathrm{dom}(\alpha)$ of α, we substitute each test candidate $t_1, \ldots, t_k$ for y in φ. For each of the resulting child nodes corresponding to $\varphi_{i,0} \equiv \varphi[t_i//y]$ of the root where $i = 1, \ldots, k$, we only substitute the test candidates that represent the current assignment α, as long as the virtual substitution is feasible. If we could define a formula ψ_i for each of those paths, we could then state in our explanation, that each

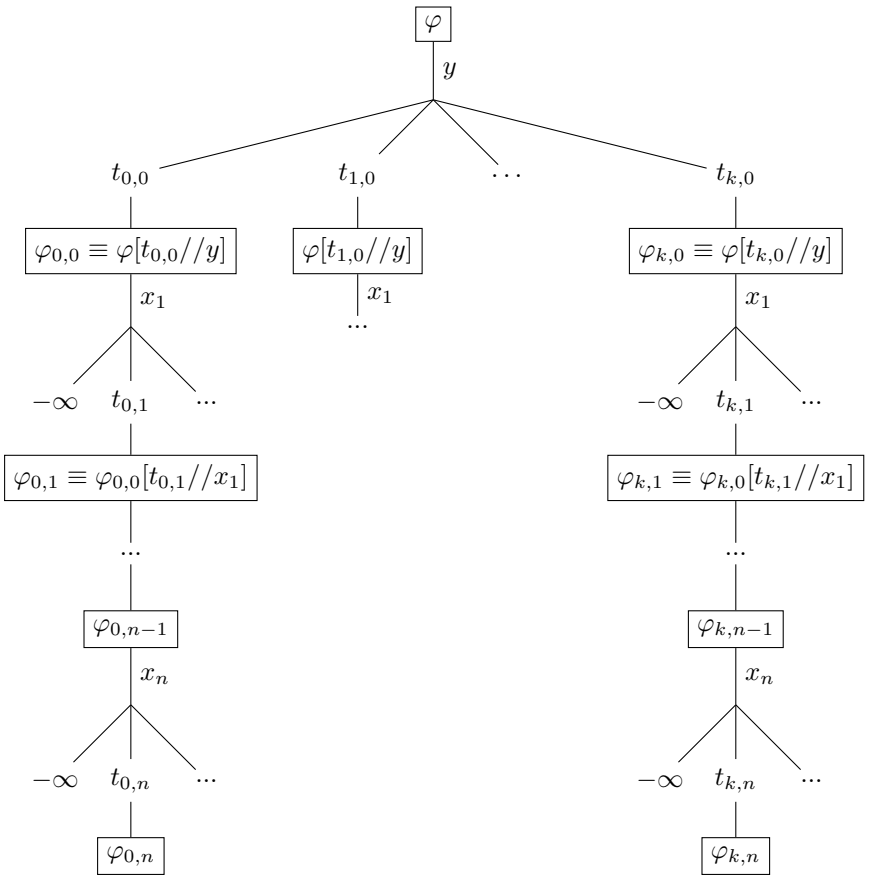

Figure 3.1: Schematic illustration of a partial virtual substitution tree

of those path descriptions implies the formula φ_{i,s_i} in the leaf node of the corresponding path. Moreover, such a description would allow us to break the procedure at any point, so that we can choose any grade between the general variant and the specialization to the conflict.

To do so, we need to tackle some challenges: First, during the generation of the virtual substitution tree, a representing test candidate $t_{j,i}$ for an assignment $\alpha(x_i)$ may not be in the set of generated test candidates $tcs(x,\varphi_{j,i-1})$, since mcSAT assigns theory variables independently from our procedure. Thus we need to consider the whole set of all representatives $rs(x,\varphi_{j,i-1})$. Second, the choice of such representatives is not unique as the representatives are symbolic and may lay on each other. Third, defining the generated path is not trivial since the regions represented by a test candidate are not cylindrically ordered. As a consequence, some more effort is needed to obtain a formula defining a sign-invariant region.

After those problems are solved, we can give an explanation lemma of the form

$$A \to \bigvee_{i=1,\ldots,k} (\psi_i \to \varphi_{i,s_i})$$

We first justify our procedure by giving theorems analogous to Theorem 2.3 that plug in a single representative and use these results afterwards to give a description of our explanation function.

Example 3.1. *We consider the formula φ and its virtual substitution tree from Example 2.1 again.*

Let the partial assignment α be given as $\alpha(x) = 3$. Then, no extension of α for y exists without making the formula conflicting. For demonstration purposes we omit computing a minimal conflicting subset of the constraints.

To eliminate y, we substitute each test candidate for y. For each of the resulting child nodes of the root, we only substitute one test candidate that represents the current assignment. In the case of the rightmost subtree, we have that $\alpha(x) = 3 > 2 + \frac{\sqrt{2}}{2}$ and therefore we substitute only $t'_2 = 2 + \frac{\sqrt{2}}{2} + \varepsilon$ for the variable x, as shown in Figure 3.2.

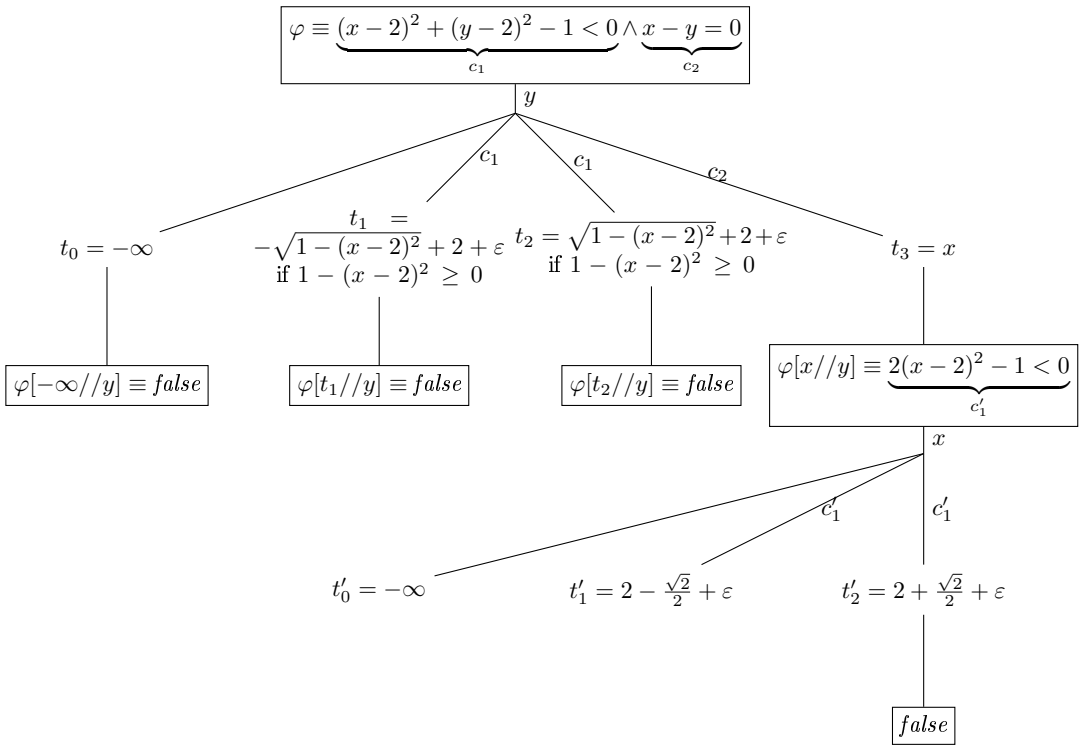

Figure 3.2: Partial virtual substitution tree

3.1 Elimination rules

Our goal is it to substitute for each variable $x_i, i = 1, \ldots, n$ only one representative for $\alpha(x_i)$. First, we need to determine this representative (as described in Section 3.1.1), then we see the relation to the pure quantifier elimination approach (in Section 3.1.2) and then we deduce the desired statements (see Section 3.1.3).

3.1.1 Determining the representative

Given a quantifier-free formula φ and a complete assignment $\alpha : \mathit{Vars}(\varphi) \to \mathbb{R}$, we need to determine a representative of the sign-invariant region in the virtual substitution tree for φ where the

current assignment $\alpha(x)$ for a variable x lays in. To do so, we order all zeros occurring in φ under the current assignment:

Definition 3.1 (Order of zeros under an assignment)**.** *Given a quantifier-free formula φ, a complete assignment $\alpha : Vars(\varphi) \to \mathbb{R}$ and a variable $x \in Vars(\varphi)$ which appears at most quadratically in φ.*

Let $Zeros(x,\varphi) = \cup_{p \in Pols(\varphi)} Zeros(x,p)$ be the set of all symbolic zeros of all polynomials of φ in x *(see Definition 2.6 for the definition of $Zeros(x,p)$). Furthermore, we define*

$$Zeros_{val}(x,\varphi,\alpha) = \{\xi \in Zeros(x,\varphi) \mid \alpha \models sc(\xi)\} = \{\xi_1, \ldots, \xi_m\}$$

as the set of the zeros with valid side conditions under α *and analogously*

$$Zeros_{inval}(x,\varphi,\alpha) = \{\xi \in Zeros(x,\varphi) \mid \alpha \not\models sc(\xi)\}$$

as the set of the zeros with invalid side conditions under α.

If $Zeros_{val}(x,\varphi,\alpha) \neq \emptyset$, we assume w.l.o.g. the order *as one of the following cases:*

$$\alpha(x) < [\![\xi_1]\!]^\alpha \sim_1 [\![\xi_2]\!]^\alpha \sim_2 \cdots \sim_{m-1} [\![\xi_m]\!]^\alpha \tag{3.1}$$

$$[\![\xi_1]\!]^\alpha \sim_1 [\![\xi_2]\!]^\alpha \sim_2 \cdots \sim_{l-1} [\![\xi_l]\!]^\alpha \sim_l \alpha(x) < [\![\xi_{l+1}]\!]^\alpha \sim_{l+1} \cdots \sim_{m-1} [\![\xi_m]\!]^\alpha \tag{3.2}$$

$$[\![\xi_1]\!]^\alpha \sim_1 [\![\xi_2]\!]^\alpha \sim_2 \cdots \sim_{m-1} [\![\xi_m]\!]^\alpha \sim_m \alpha(x) \tag{3.3}$$

where $\sim_i \in \{<, =\}$ for all $i = 1, \ldots, m$.

In the first case, the sign-invariant region of $\alpha(x)$ is the open interval $(-\infty, \xi_1)$. In the second and third cases, ξ_l respectively ξ_m is a left bound of the desired region. If $\sim_l$ is $=$, then it is a closed point interval, otherwise it is an open interval. More precisely, the set of possible representatives for an assignment is defined as follows:

Definition 3.2 (Representative of an assignment)**.** *Given a quantifier-free formula φ, a variable x which appears at most quadratically in φ and a complete assignment $\alpha \in Assigns(\varphi)$. Let $Zeros_{val}(x,\varphi,\alpha)$ be the set of all zeros of polynomials occurring in φ with valid side condition under α. A* representative of the assignment $\alpha(x)$ in x *is given as*

$$t \in Repr(x,\varphi,\alpha) \iff t = \begin{cases} -\infty & \text{if } \alpha(x) < [\![\xi]\!]^\alpha \text{ for all } \xi \in Zeros_{val}(x,\varphi,\alpha) \\ \xi & \text{if } \alpha(x) = [\![\xi]\!]^\alpha \text{ for } \xi \in Zeros_{val}(x,\varphi,\alpha) \\ \xi + \varepsilon & \text{if } [\![\xi]\!]^\alpha < \alpha(x) \text{ for } \xi \in Zeros_{val}(x,\varphi,\alpha) \text{ and there exists no} \\ & \quad \xi' \in Zeros_{val}(x,\varphi,\alpha) \text{ with } [\![\xi]\!]^\alpha < [\![\xi']\!]^\alpha \leq \alpha(x) \end{cases}$$

Note that the choice of the representative is not unique. We will see that it does not matter which one is picked. Furthermore, the ξ_l (2nd case) respectively ξ_m (3rd case) from Definition 3.1 are contained in $Repr(x,\varphi,\alpha)$, either of the form ξ or of the form $\xi + \epsilon$.

We justify that for each assignment such a representative is found and that it lays indeed in the same sign-invariant region.

Theorem 3.1. *Let φ be a quantifier-free formula and let $x \in Vars(\varphi)$ be a variable which occurs at most quadratically in φ.*

Given an assignment $\beta : Vars(\varphi) \setminus \{x\} \to \mathbb{R}$ and extensions $\alpha,\alpha' : Vars(\varphi) \to \mathbb{R}$ of β, then

$$Repr(x,\varphi,\alpha) = Repr(x,\varphi,\alpha') \implies sgn([\![p]\!]^\alpha) = sgn([\![p]\!]^{\alpha'})$$

for all $p \in Pols(\varphi)$.

Let $\alpha : Vars(\varphi) \to \mathbb{R}$ *be an assignment. For each value* $\alpha(x)$ *for a variable* x *there exists such a representative* $t \in Repr(x, \varphi, \alpha) \subseteq rs(x, \varphi)$.

Notably, given an assignment $\alpha : Vars(\varphi) \to \mathbb{R}$, *then*

$$t \in Repr(x,\varphi,\alpha) \text{ and } \alpha \models \varphi \iff \alpha \models \exists x.sc(t) \wedge x = \alpha(t) \wedge \varphi$$

Proof. The correctness of the first two statements follows directly from the correctness of the virtual substitution.

To prove the third statement, we define $\hat{\alpha} : Vars(\varphi) \to \mathbb{R}$ so that

$$\hat{\alpha}(y) = \begin{cases} \alpha(x) & x \neq y \\ \alpha(t) & x = y \end{cases}$$

By comparing the cases, it can be seen that $t \in Repr(x,\varphi,\alpha) \iff t \in Repr(x,\varphi,\hat{\alpha})$. Finally, the third statement can be proven as follows:

$$\begin{aligned} & t \in Repr(x,\varphi,\alpha) \text{ and } \alpha \models \varphi \\ \iff & t \in Repr(x,\varphi,\hat{\alpha}) \text{ and } \hat{\alpha} \models \varphi \\ \iff & \hat{\alpha} \models sc(t) \wedge x = \alpha(t) \text{ and } \hat{\alpha} \models \varphi \\ \iff & \alpha \models \exists x.sc(t) \wedge x = \alpha(t) \wedge \varphi \end{aligned}$$

□

3.1.2 Pure quantifier elimination revisited

The usual use case for virtual substitution is to eliminate a variable from a formula as given in Theorem 2.3. To show the dualities between this theorem and our extensions, we sketch how this theorem is justified.

Given a quantifier-free formula φ, we have that

$$\exists x.\varphi \equiv \bigvee_{t \in rs(x,\varphi)} (\psi_{x,\varphi,t} \wedge sc(t) \wedge \varphi[t//x])$$

where $\alpha \models \psi_{x,\varphi,t} \iff t \in Repr(x,\varphi,\alpha)$.

From Theorem 3.1 it follows that for each assignment for x there exists a test candidate that represents this value, thus

$$\bigvee_{t \in rs(x,\varphi)} \psi_{x,\varphi,t}$$

where $\alpha \models \psi_{x,\varphi,t} \iff t \in Repr(x,\varphi,\alpha)$ is a tautology and we can write

$$\exists x.\varphi \equiv \bigvee_{t \in rs(x,\varphi)} (\varphi[t//x] \wedge sc(t))$$

By applying Theorem 2.1, this results in Theorem 2.3:

$$\exists x.\varphi \equiv \bigvee_{t \in tcs(x,\varphi)} (\varphi[t//x] \wedge sc(t))$$

3.1.3 Elimination of single representatives

The observation above leads to the idea to eliminate only one test candidate according to a given assignment. Our goal is to obtain statements of the form

$$\exists x.(\varphi \wedge \psi) \equiv sc(t) \wedge \varphi[t//x] \text{ for all } t \in rs(x,\varphi)$$

where $\alpha \models \psi \iff t \in Repr(x,\varphi,\alpha)$ or weaker statements of the form

$$\exists x.(\varphi \wedge \psi) \models sc(t) \wedge \varphi[t//x] \text{ for all } t \in rs(x,\varphi)$$

where $\alpha \models \psi \implies t \in Repr(x,\varphi,\alpha)$.

The challenge is to find such a formula ψ that describes the region that is represented by a given representative.

Decomposition induced by the set of representatives

To describe a path generated by some representatives, one considers all orderings of zeros in each variable according to Definition 3.1 seen during the generation of the path. Each of those orderings induces a decomposition of the space that is made implicitly in each elimination step. As a consequence, to describe a path, one has to assure that the value of each variable lays in the same subspace according to this decomposition. In the following example, we study the properties of this decomposition.

Example 3.2. *Let φ be defined as*

$$\varphi \equiv \underbrace{x_1 > -2}_{c_1} \wedge \underbrace{x_1 < -x_2 + 2}_{c_2} \wedge (\underbrace{x_1 > x_2 + 2}_{c_3} \vee \underbrace{x_2^2 + x_1^2 < 1}_{c_4}))$$

The gray area in Figure 3.3 denotes the solutions for this problem.

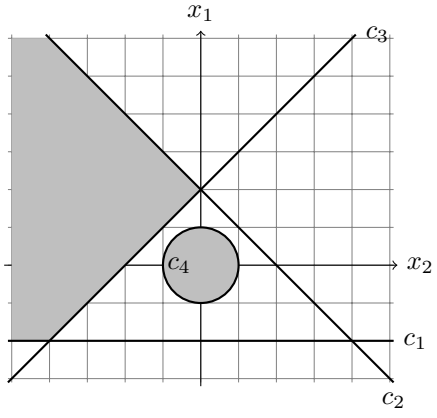

Figure 3.3: Satisfiable regions

For instance, we choose $\alpha(x_2) = \frac{3}{2}$ and $\alpha(x_1) = 0$. The partial virtual substitution tree resulting from this assignment is shown in Figure 3.4. Our goal is to define the path representing this assignment in the partial virtual substitution tree.

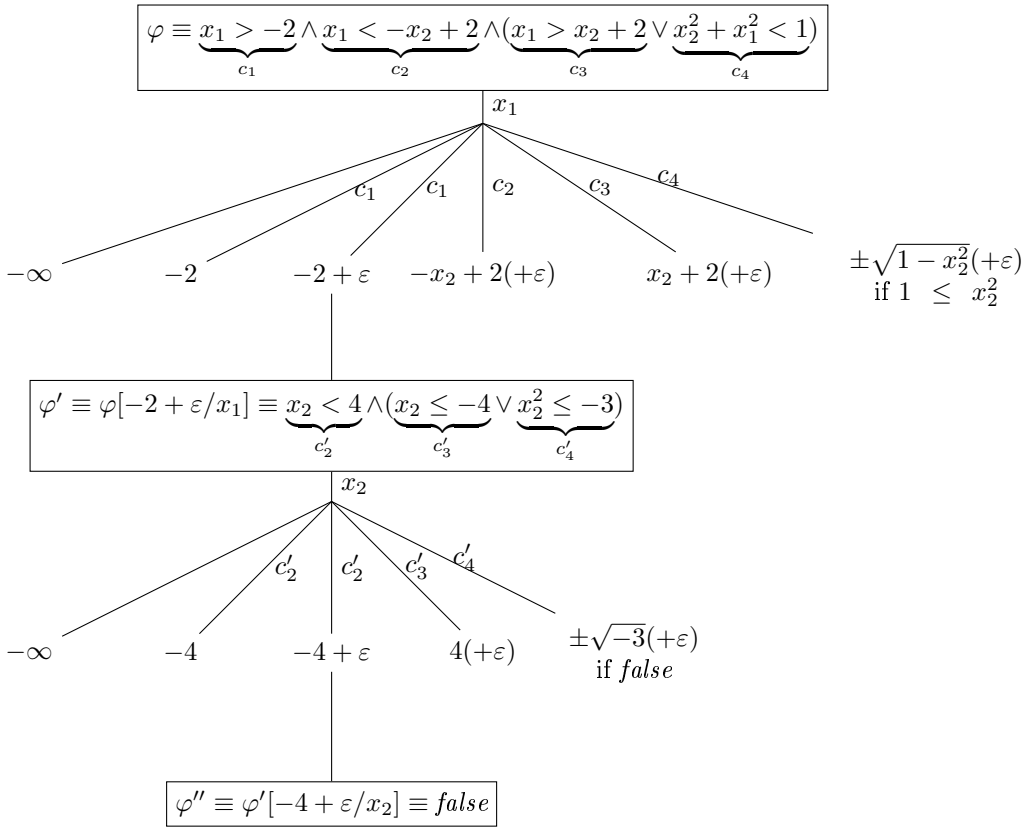

Figure 3.4: Partial virtual substitution tree

The first naive approach is to define the symbolic intervals for the eliminated variables for each step, considering only the left and right endpoints according to the current ordering of representatives:

$$-2 < x_1 < -x_2 + 2 \wedge -4 < x_2 < 4$$

Figure 3.5 shows that this region is not sign-invariant for φ.

The reasons for this are clear: When eliminating x_2, only intersection points of the polynomial c_1 with other polynomials are considered for partitioning the space (see Figure 3.6), as the representative that was substituted for x_1 originated from c_1. The intersection point of c_2 with c_3 is ignored as we are only interested in generating sign-invariant regions, which do not need to be cylindrically ordered as in a CAD. Furthermore, since the zeros of c_4 do not exist at α, it is simply ignored for defining the region in which α lays in.

The crucial consequence is that for a path defined by its representatives $t_1, \ldots t_s$, there might exist two (different) assignments α,α' represented by the path that do not induce the same order and existence of zeros in a variable. Thus the symbolic description of the path is not straight-forward.

Please note that this problem also exists if the formula is a conjunction of constraints. This can be seen by considering the similar problem

$$\underbrace{x_1 > -2}_{c_1} \wedge \underbrace{x_1 < -x_2 + 2}_{c_2} \wedge \underbrace{x_1 > x_2 + 2}_{c_3}$$

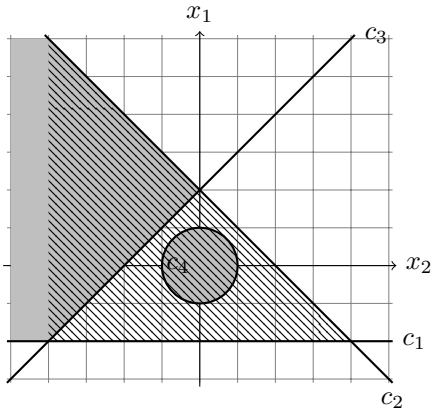

Figure 3.5: Naive definition

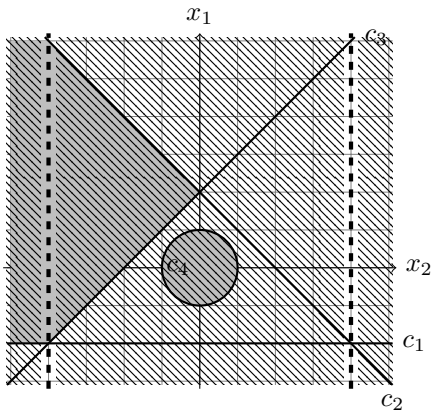

Figure 3.6: Partition into regions

respectively

$$\underbrace{x_1 > -2}_{c_1} \wedge \underbrace{x_1 < -x_2 + 2}_{c_2} \wedge \underbrace{x_2^2 + x_1^2 < 1}_{c_4}$$

The main conclusion is that the decomposition induced by the zeros in a variable is not cylindrical and thus, defining the region of a given assignment is not trivial. We propose three different formulas describing a region enclosing the current assignment $\alpha(x)$ for the current variable x in φ.

General description of the represented region

The most general form describes exactly the region represented by a representative. To make sure that the described region contains only assignments represented by the same representative t, for the case that t is of the form $\xi + \varepsilon$, we require that the value of x is greater than the value of ξ and lower than the lowest value of the possible right endpoints, or equivalently, the value of all other representatives are either lower or equal than the value of ξ or greater than the value of x:

Definition 3.3 ($\Psi_1(x,\varphi,t)$)**.** *Let φ be a formula and x a variable appearing at most quadratically in φ. Let $Zeros(x,\varphi)$ be the set of all symbolic zeros of all polynomials in φ, and let $t \in rs(x,\varphi)$. We define*

$$\Psi_1(x,\varphi,t) \quad = \quad \begin{cases} \bigwedge_{\xi \in Zeros(x,\varphi)} (sc(\xi) \to x < \xi) & \text{if} \quad t = -\infty \\ sc(\xi_r) \wedge x = \xi_r & \text{if} \quad t = \xi_r \text{ with } \xi_r \in Zeros(x,\varphi) \\ sc(\xi_r) \wedge \xi_r < x \wedge & \text{if} \quad t = \xi_r + \varepsilon \\ \bigwedge_{\xi \in Zeros(x,\varphi) \setminus \{\xi_r\}} (sc(\xi) \to (\xi \leq \xi_r \vee x < \xi)) & \text{with } \xi_r \in Zeros(x,\varphi) \end{cases}$$

Theorem 3.2. *Given a quantifier-free formula φ, a variable x occurring at most quadratically in φ, $\alpha : Vars(\varphi) \to \mathbb{R}$ and $t \in Repr(x,\varphi,\alpha)$. Then*

$$\alpha \models \Psi_1(x,\varphi,t)$$

and

$$\alpha' \models \Psi_1(x,\varphi,t) \iff t \in Repr(x,\varphi,\alpha')$$

for all $\alpha' : Vars(\varphi) \to \mathbb{R}$.

In other words, this means that $\alpha : Vars(\varphi) \to \mathbb{R}$ lays in the same maximal sign-invariant region as any $\alpha' \in Assigns(\varphi)$ that satisfies $\Psi_1(x,\varphi,t)$.

Proof. The second statement follows by comparing Definition 3.2 with the definition of Ψ_1.
From Theorem 3.1 and the second statement it follows the first statement. □

Example 3.3. *Let us consider Example 3.2 from above again. If we would choose $\alpha(x_2) = \frac{1}{2}, \alpha(x_1) = -\frac{3}{2}$, we would obtain the formula*

$$\begin{aligned} & -2 < x_1 \\ \wedge & (-x_2 + 2 \leq -2 \vee x_1 < -x_2 + 2) \\ \wedge & (x_2 + 2 \leq -2 \vee x_1 < x_2 + 2) \\ \wedge & 1 \leq x_2^2 \to (-\sqrt{1 - x_2^2} \leq -2 \vee x_1 < -\sqrt{1 - x_2^2}) \\ \wedge & 1 \leq x_2^2 \to (\sqrt{1 - x_2^2} \leq -2 \vee x_1 < \sqrt{1 - x_2^2}) \\ \wedge & -4 < x_2 < 4 \end{aligned}$$

The result is the dashed region shown in Figure 3.7.

Describing the lower and upper bounds of the current interval

So far we only fixed the lower bound of the (symbolic) interval the current assignment lays in. In this variant, we also determine the upper bound for the case that the current assignment lays in an open interval.

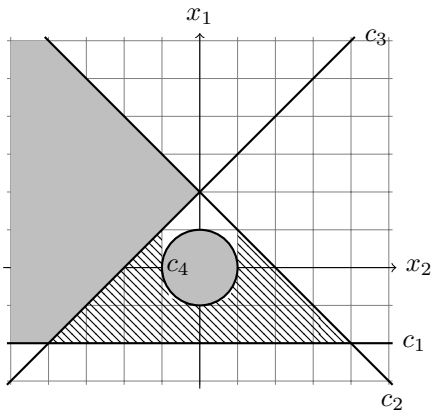

Figure 3.7: First variant

Definition 3.4 ($\Psi_2(x,\varphi,t,\alpha)$)**.** *Let φ be a formula and x a variable appearing at most quadratically in φ. Let the order of all zeros be defined as in Definition 3.1, and let $t \in rs(x,\varphi)$. For the second case (see Equation 3.2), we set $r = l$ and for the third case (see Equation 3.3), we set $r = m$. In both of these cases, for $\sim_r$ equals "$=$", we assume $t = \xi_r$, for $\sim_r$ equals $<$, we assume $t = \xi_r + \varepsilon$. We define*

$$\Psi_2(x,\varphi,t,\alpha) = \begin{cases} \bigwedge_{\xi \in Zeros(x,\varphi)} \neg sc(\xi) & \text{if } t = -\infty \text{ and } Zeros_{val}(x,\varphi,\alpha) = \emptyset \\ sc(\xi_1) \wedge x < \xi_1 \wedge \bigwedge_{\xi \in Zeros(x,\varphi)\setminus\{\xi_1\}} (sc(\xi) \rightarrow \xi_1 \leq \xi) & \text{if } t = -\infty \text{ and } Zeros_{val}(x,\varphi,\alpha) \neq \emptyset \\ sc(\xi_r) \wedge x = \xi_r & \text{if } t = \xi_r \text{ in the 2nd (Eq. 3.2) or 3rd case (Eq. 3.3)} \\ sc(\xi_r) \wedge \xi_r < x \wedge sc(\xi_{r+1}) \wedge x < \xi_{r+1} \wedge \bigwedge_{\xi \in Zeros(x,\varphi)\setminus\{\xi_r,\xi_{r+1}\}} (sc(\xi) \rightarrow (\xi \leq \xi_r \vee \xi_{r+1} \leq \xi)) & \text{if } t = \xi_r + \varepsilon \text{ in the 2nd case (Eq. 3.2)} \\ sc(\xi_r) \wedge \xi_r < x \wedge \bigwedge_{\xi \in Zeros(x,\varphi)\setminus\{\xi_r\}} (sc(\xi) \rightarrow (\xi \leq \xi_r)) & \text{if } t = \xi_r + \varepsilon \text{ in the 3rd case (Eq. 3.3)} \end{cases}$$

Theorem 3.3. *Given a quantifier-free formula φ, a variable x occurring at most quadratically in φ, $\alpha : Vars(\varphi) \rightarrow \mathbb{R}$ and $t \in Repr(x,\varphi,\alpha)$. Then*

$$\alpha \models \Psi_2(x,\varphi,t,\alpha)$$

and

$$\alpha' \models \Psi_2(x,\varphi,t,\alpha) \implies t \in Repr(x,\varphi,\alpha')$$

for all $\alpha' : Vars(\varphi) \rightarrow \mathbb{R}$.

Proof. The validity of the first statement can be seen by comparing the region defined in each case with the assignments generating the formula.

By comparing each case, it follows that $\Psi_2(x,\varphi,t,\alpha) \implies \Psi_1(x,\varphi,t,\alpha)$ and therefore by the correctness of $\Psi_1(x,\varphi,t,\alpha)$ it follows the second statement. □

Example 3.4. *As for the preceding formula, we consider the assignment* $\alpha(x_2) = \frac{1}{2}, \alpha(x_1) = -\frac{3}{2}$ *in Example 3.2:*

$$\begin{aligned}
&-2 < x_1 \wedge x_1 < -\sqrt{1-x_2^2} \wedge 1 \leq x_2^2 \\
&\wedge(-x_2 + 2 \leq -2 \vee -\sqrt{1-x_2^2} \leq -x_2 + 2) \\
&\wedge(x_2 + 2 \leq -2 \vee -\sqrt{1-x_2^2} \leq x_2 + 2) \\
&\wedge 1 \leq x_2^2 \rightarrow (\sqrt{1-x_2^2} \leq -2 \vee -\sqrt{1-x_2^2} \leq \sqrt{1-x_2^2}) \\
&\wedge -4 < x_2 < 4
\end{aligned}$$

The result is the dashed region shown in Figure 3.8.

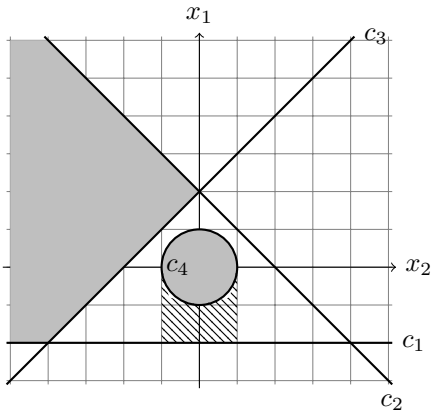

Figure 3.8: Second variant

Describing the order of all zeros

In the strongest formula we propose, we define the total order of all zeros in the variable x.

Definition 3.5 ($\Psi_3(x,\varphi,t,\alpha)$)**.** *Let φ be a formula and x a variable appearing at most quadratically in φ. Let the order of all zeros be defined as in Definition 3.1, and let $t \in rs(x,\varphi)$. For the second case (see Equation 3.2), we set $r = l$ and for the third case (see Equation 3.3), we set $r = m$. In both of these cases, for $\sim_r$ equals " = ", we assume $t = \xi_r$, for $\sim_r$ equals $<$, we assume $t = \xi_r + \varepsilon$. We define*

$$scs \equiv \bigwedge_{\xi \in Zeros_{val}(x,\varphi,\alpha)} sc(\xi) \wedge \bigwedge_{\xi \in Zeros_{inval}(x,\varphi,\alpha)} \neg sc(\xi)$$

and

$$\Psi_3(x,\varphi,t,\alpha) = \begin{cases} scs & \text{if } t = -\infty \text{ and } Zeros_{val}(x,\varphi,\alpha) = \emptyset \\ x < \xi_1 \\ \wedge \bigwedge_{i\in\{1,\dots,m-1\}}(\xi_i \sim_i \xi_{i+1}) \wedge scs & \text{if } t = -\infty \text{ and } Zeros_{val}(x,\varphi,\alpha) \neq \emptyset \\ x = \xi_r & \text{if } t = \xi_r \text{ in the 2nd (Eq. 3.2) or} \\ \wedge \bigwedge_{i\in\{1,\dots,m-1\}}(\xi_i \sim_i \xi_{i+1}) \wedge scs & \text{3rd case (Eq. 3.3)} \\ \xi_r < x \wedge x < \xi_{r+1} \wedge & \text{if } t = \xi_r + \varepsilon \text{ in the 2nd case} \\ \bigwedge_{i\in\{1,\dots,m-1\}\setminus\{r\}}(\xi_i \sim_i \xi_{i+1}) \wedge scs & \text{(Eq. 3.2)} \\ \xi_r < x \wedge & \text{if } t = \xi_r + \varepsilon \text{ in the 3rd case} \\ \bigwedge_{i\in\{1,\dots,m-1\}}(\xi_i \sim_i \xi_{i+1}) \wedge scs & \text{(Eq. 3.3)} \end{cases}$$

Theorem 3.4. *Given a quantifier-free formula φ, a variable x occurring at most quadratically in φ, $\alpha : Vars(\varphi) \to \mathbb{R}$ and $t \in Repr(x,\varphi,\alpha)$. Then*

$$\alpha \models \Psi_3(x,\varphi,t,\alpha)$$

and

$$\alpha' \models \Psi_3(x,\varphi,t,\alpha) \implies t \in Repr(x,\varphi,\alpha')$$

for all $\alpha' : Vars(\varphi) \to \mathbb{R}$.

Proof. The validity of the first statement can be seen by comparing the region defined in each case with the assignments generating the formula.

By comparing each case, it follows that $\Psi_3(x,\varphi,t,\alpha) \implies \Psi_1(x,\varphi,t,\alpha)$ and therefore by the correctness of $\Psi_1(x,\varphi,t,\alpha)$ follows the second statement. □

Example 3.5. *Again, we consider Example 3.2 and the assignment $\alpha(x_2) = \frac{1}{2}, \alpha(x_1) = -\frac{3}{2}$:*

$$\begin{aligned} &-2 < x_1 \wedge x_1 < -\sqrt{1-x_2^2} \wedge 1 \leq x_2^2 \\ \wedge &-\sqrt{1-x_2^2} < \sqrt{1-x_2^2} < -x_2 + 2 < x_2 + 2 \\ \wedge &-4 < x_2 < 4 \end{aligned}$$

The result is the dashed region shown in Figure 3.9. Interestingly, all intersection points of all polynomials are considered, as well as the existence of the zeros is fixed. As a result, the described regions are cylindrically ordered.

Substituting single test candidates

The proposed formulas allow us to substitute single test candidates as given by the following theorem:

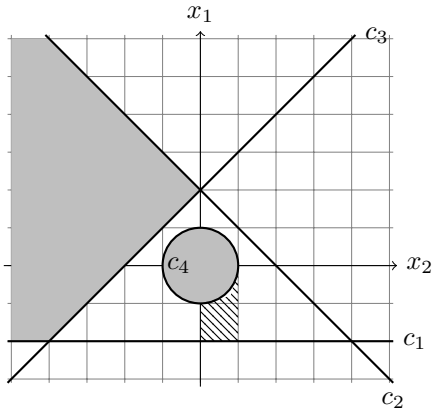

Figure 3.9: Third variant

Theorem 3.5. *Given a quantifier-free formula φ and a variable x occurring at most quadratically in φ. Then*

$$\exists x.\varphi \wedge \Psi_1(x,\varphi,t) \equiv sc(t) \wedge \varphi[t//x]$$
$$\exists x.\varphi \wedge \Psi_2(x,\varphi,t,\alpha) \models sc(t) \wedge \varphi[t//x]$$
$$\exists x.\varphi \wedge \Psi_3(x,\varphi,t,\alpha) \models sc(t) \wedge \varphi[t//x]$$

for all $\alpha : Vars(\varphi) \to \mathbb{R}$ and $t \in Repr(x,\varphi,\alpha)$.

Proof. The first statement is proven as follows:

Let $\alpha : Vars(\varphi) \to \mathbb{R}$ and $t \in Repr(x,\varphi,\alpha)$. Let $\alpha' : Vars(\varphi) \setminus \{x\} \to \mathbb{R}$. Then

$$\begin{aligned}
&\alpha' \models \exists x.\varphi \wedge \Psi_1(x,\varphi,t) \\
\iff &\text{there exists an extension } \alpha'_x : Vars(\varphi) \to \mathbb{R} \text{ of } \alpha' \text{ with } \alpha'_x \models \varphi \wedge \Psi_1(x,\varphi,t) \\
\overset{\text{Theorem 3.2}}{\iff} &\text{there exists an extension } \alpha'_x : Vars(\varphi) \to \mathbb{R} \text{ of } \alpha' \text{ with } t \in Repr(x,\varphi,\alpha'_x) \text{ and } \alpha'_x \models \varphi \\
\overset{\text{Theorem 3.1}}{\iff} &\text{there exists an extension } \alpha'_x : Vars(\varphi) \to \mathbb{R} \text{ of } \alpha' \text{ with } \alpha'_x \models sc(t) \wedge x = \alpha'(t) \text{ and } \alpha'_x \models \varphi \\
\iff &\alpha' \models sc(t) \wedge \varphi[\alpha'(t)/x] \\
\overset{\text{Theorem 2.2}}{\iff} &\alpha' \models sc(t) \wedge \varphi[t//x]
\end{aligned}$$

From Theorem 3.2, 3.3 and 3.4 it follows that

$$\Psi_2(x,\varphi,t,\alpha) \models \Psi_1(x,\varphi,t)$$

respectively

$$\Psi_3(x,\varphi,t,\alpha) \models \Psi_1(x,\varphi,t)$$

for each $t \in Repr(x,\varphi,\alpha)$ and $\alpha : Vars(\varphi) \to \mathbb{R}$. The validity of the second and third statements follows. □

Representation of zeros

Note that the symbolic representation of a zero of a polynomial is only given as a square root expression as specified in Definition 2.5, which is in general not an arithmetic expression. We will examine the resulting requirements on the solver later.

3.2 Explanation function

Given a conjunction A of constraints from $\mathbb{R}[x_1,\ldots,x_n,y]$ such that y occurs at most quadratically in each constraint, assigned variables $x_1,\ldots,x_n$ with assignment α so that every extension of α for y makes A conflicting, we construct the virtual substitution tree as follows:

- The root node contains $\varphi \equiv A$ where y occurs at most quadratically in φ.
- Let $\tilde{t}_1,\ldots,\tilde{t}_k \in tcs(y,\varphi)$ be the test candidates for y. For each test candidate $\tilde{t}_i$, a child node is created containing the formula $\varphi_{i,0} \equiv \varphi[\tilde{t}_i//y] \wedge sc(\tilde{t}_i)$.
- For each of these child nodes, only one path representing α is generated as follows:
  - Starting with $j = 0$, as long as there exists a variable in $\varphi_{i,j}$, it is eliminated:
    - Let $x_{i,j+1} \in Vars(\varphi_{i,j})$ be the variable that has been assigned last.
    - If the degree of $x_{i,j+1}$ in $\varphi_{i,j}$ is greater than 2, then stop here.
    - Let $\{\xi_1,\ldots,\xi_m\} = Zeros(\varphi_{i,j})$ be all symbolic zeros of all polynomials in $\varphi_{i,j}$. They are ordered w.l.o.g. according to α:

      $$[\![\xi_1]\!]^\alpha \leq \cdots \leq [\![\xi_m]\!]^\alpha$$

      A representative $t_{i,j+1} \in Repr(x_{i,j+1},\varphi_{i,j},\alpha)$ is determined using this order.
    - A child node is created containing the formula

      $$\varphi_{i,j+1} \equiv \varphi_{i,j}[t_{i,j+1}//x_{i,j+1}] \wedge sc(t_{i,j+1})$$

  - Let n_i denote the number of elimination steps for $\tilde{t}_i$, i.e. the maximal value considered for j above. Note that n_i is less or equal the number of variables in $\varphi_{i,0}$, but in general not equal.

For each test candidate $\tilde{t}_i$ for y we define the formula

$$\omega_i \equiv \left(\bigwedge_{j=1,\ldots,s_i} \Psi(x_{i,j},\varphi_{i,j-1},t_{i,j},\alpha) \right) \rightarrow \varphi_{i,s_i}$$

where $\Psi \in \{\Psi_1,\Psi_2,\Psi_3\}$ (for Ψ_1 we omit the last argument α) and $0 \leq s_i \leq n_i$. For $s_i = 0$ the conjunction on the left-hand side of the implication is defined as *true*. Then the formula ω_i describes the path defined by α and the choice "$y = \tilde{t}_i$".

As a result, the explanation clause is defined as

$$explain(A,\alpha) \equiv A \rightarrow \bigvee_{i=1,\ldots,k} \omega_i$$

A pseudo-code algorithm for computing $explain(A,\alpha)$ is given in Figure 3.10.

In the following, we denote the order on the variables in which they are eliminated as *elimination order*.

EXPLAIN(A, α)
1 **for** $\tilde{t}_i \in tcs(y,A)$
2 **do** $\omega_i \leftarrow constructPath(A[\tilde{t}_i//y] \wedge sc(\tilde{t}_i), \alpha)$
3 **return** $\bigvee_{\tilde{t}_i \in tcs(y,A)} \omega_i$

CONSTRUCTPATH(φ, α)
1 $R \leftarrow true$
2 **while** $existsVarIn(\varphi)$
3 **do** $x \leftarrow lastAssignedVariableIn(\varphi)$
4 **if** *degree of* φ *in* x *is greater than* 2
5 **then break**
6 $t \leftarrow$ *chose one from* $Repr(x, \varphi, \alpha)$
7 **if** $breakHeuristic(x,\varphi,t,\alpha)$
8 **then break**
9 $R \leftarrow R \wedge \Psi(x, \varphi, t, \alpha)$
10 $\varphi \leftarrow \varphi[t//x] \wedge sc(t)$
11 **return** $R \rightarrow \varphi$

where $\Psi \in \{\Psi_1, \Psi_2, \Psi_3\}$.

Figure 3.10: Basic algorithm

3.2.1 Requirements on the mcSAT solver

As we express zeros of polynomials as square root expressions in *explain*(A, α), the solver needs to be able to handle them for evaluating constraints.

Moreover, we require the solver to make sure that constraints containing a square root expression do not occur as part of an input to *explain* for several reasons. First, we did not define yet how we handle square root expressions. And more important, to prove termination, we assume that all constraints in the input are either from the original formula φ or have been generated by virtual substitution using the same elimination order as induced by the current order in which the variables have been assigned. If we also allow the latter, a consequence is that the solver needs to assign all variables in a fixed order. Nevertheless, it is possible to perform theory assignments, decisions and propagations in such a way that fulfills this requirement.

Furthermore, the generated lemma is not a clause in general. However, the mcSAT implementation of SMT-RAT [Cor16] can handle these expressions.

We do not go into details how these issues are solved as it goes beyond the scope of this thesis.

3.2.2 Correctness

Theorem 3.6. *The explanation lemma* $explain(A, \alpha)$ *fulfills the requirement* $\models explain(A, \alpha)$.

Proof. From Theorem 3.5 it follows

$$\varphi_{i,0} \models \exists x_{i,1} \ldots \exists x_{i,s_i}.\varphi_{i,0} \models \left(\bigwedge_{j=1,\ldots,s_i} \Psi(x_{i,j},\varphi_{i,j-1},t_{i,j},\alpha) \right) \rightarrow \varphi_{i,s_i}$$

for $\Psi \in \{\Psi_1,\Psi_2,\Psi_3\}$, all $i = 1,\ldots,k$ and all $0 \leq s_i \leq n_i$.

By applying Theorem 2.3 we get

$$A \models \exists y.A \equiv \bigvee_{i=1,\ldots,k} (A[\tilde{t}_i//y] \wedge sc(\tilde{t}_i)) \models \bigvee_{i=1,\ldots,k} \omega_i$$

Thus, $\mathit{explain}(A,\alpha)$ is a tautology. □

Theorem 3.7. *The explanation lemma $\mathit{explain}(A,\alpha)$ of the form $A \rightarrow \hat{\varphi}$ for some formula $\hat{\varphi}$ excludes the current assignment under the assumption of A, e.g. $\alpha \not\models \hat{\varphi}$.*

Proof. To be proven is that $\alpha \not\models (\bigvee_{i=1,\ldots,k} \omega_i)$, or equivalently,

$$\alpha \not\models \omega_i \equiv \left(\left(\bigwedge_{j=1,\ldots,s_i} \Psi(x_{i,j},\varphi_{i,j-1},t_{i,j},\alpha) \right) \rightarrow \varphi_{i,s_i} \right)$$

for all $i = 1,\ldots,k$.

From Theorem 3.2, 3.3 and 3.4 it follows that $\alpha \models \Psi(x_{i,j},\varphi_{i,j-1},t_{i,j},\alpha)$ for $\Psi \in \{\Psi_1,\Psi_2,\Psi_3\}$ and all $0 \leq j \leq n_i$.

From $\alpha \not\models \exists y.\varphi$ and Theorem 2.3 it follows that $\alpha \not\models \varphi_{i,0}$. Because $t_{i,j} \in \mathit{Repr}(x_{i,j},\varphi_{i,j-1},\alpha)$ for $0 \leq j \leq n_i$, we have that

$$\begin{aligned}
&\exists x_{i,j}.\varphi_{i,j-1} \wedge x_{i,j} = \alpha(x_{i,j}) \\
\overset{\text{Theorem 3.1}}{\equiv}\ &\exists x_{i,j}.\varphi_{i,j-1} \wedge x_{i,j} = \alpha(t_{i,j}) \\
\equiv\ &\varphi_{i,j-1}[\alpha(t_{i,j})/x_{i,j}] \wedge sc(t_{i,j}) \\
\overset{\text{Theorem 2.2}}{\equiv}\ &\varphi_{i,j-1}[t_{i,j}//x_{i,j}] \wedge sc(t_{i,j}) \\
\equiv\ &\varphi_{i,j}
\end{aligned}$$

for $0 \leq j \leq n_i$. Per induction follows that $\alpha \not\models \varphi_{i,j}$ for $0 \leq j \leq n_i$.

Thus, $\alpha \not\models \omega_i$ for all $i = 1,\ldots,k$. □

Assuming the restriction to the solver from above, we can guarantee termination by giving a finite basis:

Theorem 3.8. *The explanation lemma $\mathit{explain}(A,\alpha)$ only contains constraints from a finite basis $\mathbb{B}$ under the assumption that only constraints from the original formula and constraints obtained by virtual substitution using the same elimination order are contained in A.*

Proof. Given the initial input formula φ with variables $\mathit{Vars}(\mathcal{C}) = \{x_1,\ldots,x_n\}$ and an elimination order on the variables $x_1 < \cdots < x_n$, we generate the closure C under virtual substitution:

- $C_0 := \mathit{Constraints}(\varphi)$
- $C_i := C_{i-1} \cup \mathit{Subst}(C_{i-1})$
- $C := C_i$ where i is the smallest index such that $C_i = C_j$ for all $j > i$

A *substitution step* $\mathit{Subst}(S)$ for a set of constraints S is the substitution of one representative from a constraint in S into a constraint in S. More precisely, it is defined as follows:

- Let $l_1, l_2 \in S$ with $x_i \in \mathit{Vars}(l_1) \cap \mathit{Vars}(l_2)$ occurring at most quadratically in l_1 and l_2 and $\mathit{Vars}(l_1), \mathit{Vars}(l_2) \subseteq \mathit{Vars}(S) \setminus \{x_1,\ldots,x_{i-1}\}$ and $t \in rs(x_i,l_1)$ (where x_i, l_1, l_2, t have not been chosen together before)

- $s := l_2[t//x_i] \wedge sc(t)$
- $Subst(S) := Constraints(s)$

We proof that C indeed exists. Observe that:

1. Given a set of constraints S, then $Vars(Subst(S)) \subseteq Vars(S) \setminus \{x\}$ where x is the eliminated variable in the substitution step $Subst(S)$, because each virtual substitution rule results in a formula that does not contain the eliminated variable (see correctness proof of virtual substitution).
2. The (finite) set of initial variables is given as $Vars(C_0)$. We have that $Vars(C_i) = Vars(C_0)$ for all $i \in \mathbb{N}_0$.
3. From S is finite follows $Subst(S)$ is finite (see correctness proof of virtual substitution). C_0 is finite, and therefore C_i for all $i \in \mathbb{N}_0$ is finite. Let $s_{max} := \max\{|Subst(C_i)| \mid i \in \mathbb{N}_0\}$.

The *set of constraints $L_{i,k}$ containing the variable x_k in C_i* is given as

$$L_{i,k} := \{l \in C_i \mid x_k \in Vars(l)\}$$

for $k = 1, \ldots, n$. For $k = n+1$ we set $L_{i,n+1} := \{l \in C_i \mid Vars(l) = \emptyset\}$.

An upper bound for this set can be given as

$$|L_{i,k}| \leq |L_{0,k}| + \sum_{m,n<k} |L_{i,m} + L_{i,n}|! * s_{max} \qquad \text{for all } i \in \mathbb{N}_0, k = 1, \ldots, n \tag{3.4}$$

because either such a constraint $l \in L_{i,k}$ is contained in C_0 or is part of an elimination result. Because of the choice of l_1 and l_2 in a substitution step and Observation 1, those are the constraints containing a variable with lower elimination order.

We proof the hypothesis

$$\text{there exists an } i \geq 0 \text{ so that} |L_{i,k}| = |L_{j,k}| \text{ for all } j \geq i \tag{3.5}$$

for all $k = 1, \ldots, n+1$ by induction over k.

Basis: From Observation 2 and the bound given in Equation 3.4 it follows

$$L_{j,0} = L_{0,0} \text{ for all } j \geq 0 \tag{3.6}$$

Inductive step: Let Equation 3.5 be valid for all $k < l$. Let $h := \max\{i \mid |L_{i,k}| = |L_{j,k}| \text{ for all } j \geq i, k < l\}$. Then

$$|L_{h,l}| \leq |L_{0,l}| + \sum_{m,n<l} |L_{h,m} + L_{h,n}|! * s_{max}$$

Because Observation 3 says that the substitution gives a finite set of new constraints, there exists an i so that $|L_{i,l}| = |L_{j,l}|$ for all $j > i$.

We proved that Equation 3.5 is valid for $k = 1, \ldots, n+1$ and thus $|L_{i,k}|$ is finite for $k = 1, \ldots, n+1$ and for an $i \in \mathbb{N}_0$. Because all constraints in $|L_{i,n+1}|$ do not contain any variables, they cannot generate new constraints and thus we have $C = \cup_{k=1,\ldots,n+1} L_{i,k}$. It follows that C is finite.

Let C' denote the union of all closures obtained with each possible elimination order. Then the explanations contain constraints from C' and equations over the zeros of polynomials from C'. Altogether, the finite basis is given as:

$$\mathbb{B} = C'$$
$$\cup \{p_1 \sim p_2 \mid p_1, p_2 \in \mathit{Vars}(C') \cup \bigcup_{x \in \mathit{Vars}(C')} \mathit{Zeros}(x, C'), \sim \in \{=, \neq, <, >, \leq, \geq\}\}$$

□

3.2.3 Dealing with representatives left out by virtual substitution

One strength of the virtual substitution to test a formula for satisfiability is that some representatives can be left out as shown by Theorem 2.1. Unfortunately, we cannot exploit this observation in the same way in our method. Substituting a representative that is not a test candidate into its originating constraint gives *false* (if the relation symbol is one of $\neq, =, <, >$). However, because we assume an arbitrary structure of the formula containing $\wedge$ and $\vee$, the whole formula may not become *false* and thus, we have to proceed with our procedure as with any other representative.

However, substituting such a representative into the formula is not desirable, as such a representative would never be considered by virtual substitution. So we decide that if we detect that $t_{i,j}$ can be chosen so that $t_{i,j} \notin \mathit{tcs}(x_{i,j}, \varphi_{i,j-1})$, we stop the procedure at the node containing $\varphi_{i,j-1}$, e.g. we set $s_i = j - 1$. The corresponding break heuristic is shown in Figure 3.11.

```
BREAKHEURISTIC(x, φ, t, α)
1  return ¬(Repr(x, φ, α) ⊆ tcs(x,φ))
```

Figure 3.11: Break heuristic

Furthermore, from the observation above we conclude:

Theorem 3.9. *Let φ be a conjunction of constraints. Given a representative $t \in rs(x, p \sim 0) \setminus \mathit{tcs}(x,\varphi)$ for a constraint $p \sim 0 \in \mathit{Constraints}(\varphi)$ with $\sim \in \{<, >, =, \neq\}$, then*

$$\varphi[t//x] \equiv \mathit{false}$$

Proof. $(p \sim 0)[t//x] \equiv \mathit{false}$ can be seen by plugging in the respective representatives into the formula and applying the corresponding rules. □

3.2.4 Choice of representatives is not unique

As already mentioned, the choice of the generated path in the virtual substitution tree is not unique, i.e. an assignment can be represented by multiple representatives. This is caused by the fact that two or more symbolic zeros may lay on each other under a given assignment. In these cases, we can pick any representative representing the assignment, because we did not assume a specific representative in the proofs of the Theorems 3.2, 3.3, 3.4 and 3.5.

Example 3.6. *Let the substitution tree in Figure 3.12 be a subtree of an input after eliminating the unassigned variable y. Assume the assignment $\alpha(x_1) = 1, \alpha(x_2) = 0, \alpha(x_3) = 0$.*

We can observe that all possible representatives behave the same:

- All possible paths representing the current assignment lead to a leaf containing *false*.

- The structure is the same: Let x be a variable with $|Repr(x,\varphi,\alpha)| > 1$. Let x' be a variable eliminated after x has been eliminated. Then $\alpha(x')$ falls either into a point interval (i.e. all of its possible representatives are zeros ξ) or into an open interval (i.e. all of its possible representatives are zeros plus an infinitesimal $\xi + \varepsilon$) independently from the choice of the representative $t \in Repr(x,\varphi,\alpha)$.

Note that some choices $t \in Repr(x,\varphi,\alpha)$ may result in shorter subtrees than other choices, because the substitution results are syntactically different.

3.2.5 Elimination order

Although the elimination order is not relevant to the correctness of the method, it has an influence on the effectiveness of the generated explanation.

To understand the effect of an explanation, consider how the framework resolves conflicts: The returned explanation deflects the current conflict, which means that after receiving an explanation, the mcSAT solver reverts at least one theory variable assignment to fix the infeasible solver state. This is also called *backtracking*. After backtracking, the solver continues searching for solutions, i.e. performing Boolean search and assigning values to theory variables.

In the following let $i \in \{1,\dots,k\}$. Let the variables $x_{i,1},\dots,x_{i,n}$ be eliminated according to the order $x_{i,1} < x_{i,2} < \cdots < x_{i,n}$ when constructing the partial virtual substitution tree. The formula $\Psi \in \{\Psi_1, \Psi_2, \Psi_3\}$ defining the region of a variable $x_{i,k}$ depends on the variables $x_{i,k+1},\dots,x_{i,n}$ eliminated later and $x_{i,k}$ itself. Likewise the formula in the corresponding node $\varphi_{i,k}$ depends on $x_{i,k+1},\dots,x_{i,n}$, hence ω_i is of the form:

$$\underbrace{\Psi(x_{i,1},\varphi_{i,0},t_{i,1},\alpha)}_{in\ x_{i,1},\dots,x_{i,n}} \wedge \underbrace{\Psi(x_{i,2},\varphi_{i,1},t_{i,2},\alpha)}_{in\ x_{i,2},\dots,x_{i,n}} \wedge \underbrace{\Psi(x_{i,s_i},\varphi_{i,s_i-1},t_{i,s_i},\alpha)}_{in\ x_{i,s_i},\dots,x_{i,n}} \rightarrow \underbrace{\varphi_{i,s_i}}_{in\ x_{i,s_i+1},\dots,x_{i,n}}$$

where x_{i,s_i} is the variable eliminated last.

It is desirable that the generated explanation keeps conflict resolution simple. If we would eliminate the variables in the same order as mcSAT assigns it, then the first variable to be backtracked would be $x_{i,n}$. Backtracking and finding a suiting value for $x_{i,n}$ is expensive because all constraints in ω_i become univariate. The reason for this is clear: The dependencies between the variables are destroyed, since the definition $\psi(x_{i,1},\dots,x_{i,n})$ of $x_{i,1}$ being the first assigned variable depends on all other variables, but the framework assigned a value to $x_{i,1}$ independently from all other variables. We need to preserve the dependencies between the variables, this is why we eliminate the variables in the reverse order in which they have been assigned. This way, when backtracking $x_{1,n}$, only a single constraint in ω_i becomes univariate and allows much more lightweight "guessing" of values for theory variables.

3.2.6 Variables involved in a conflict

Note that if all variables $x_{i,j+1}, j = 0,\dots,n_i$ occur at most quadratically in $\varphi_{i,j}$, then $\varphi_{i,n_i} \equiv \mathit{false}$ for $i = 1,\dots,k$, because the given assignment α is conflicting with A. As we eliminate all variables occurring in the input formula φ, all constraints in φ_{i,n_i} for $i = 1,\dots,k$ are constants and we get indeed syntactically "false".

Furthermore, it is possible that not all variables occurring in $\varphi_{i,0}$ are eliminated as they may disappear during elimination of another variable. This is caused by the fact that when selecting a test candidate $\tilde{t}_i$ for y, only a few variables may cause unsatisfiability for $\varphi_{i,0} \equiv \varphi[\tilde{t}_i//y]$. The infeasibility tells us only that for each assigned variable $x = x_1,\dots,x_n$ there exists a test candidate $\tilde{t}_i$ so that x is part of the reason for the unsatisfiability of $\varphi[\tilde{t}_i//y]$.

Example 3.7. *Consider the formula*

$$\varphi \equiv c_1 : x_1 < y \wedge c_2 : x_2^2 < y^2 \wedge c_3 : y < x_3$$

together with the assignment

$$x_1 \mapsto -1, x_2 \mapsto 1, x_3 \mapsto 1$$

The resulting situation after plugging in the assignments into the formula is given in Figure 3.13. One can see that all constraints are necessary to produce the conflict, and that given a value for y, not all constraints cause the unsatisfiability for this value.

The virtual substitution tree in Figure 3.14 shows that the fourth subtree depends only on variables x_2 and x_3, because they are the only variables substituted or appearing in a chosen representative. For the second and third subtree, it can be shown that it is sufficient to eliminate only variables x_1 and x_2 to obtain a leaf node containing false. Thus, which variables are skipped during elimination depends also on the order in which they are eliminated.

3.2.7 Generation of partial paths

As already seen, we can stop the generation of the partial substitution tree at any node and generate a formula describing the partial path and the formula at the end of this path. This allows us to vary between generality and specialization to the conflict. For example, for $s_i = 0$ for all $i = 1, \ldots, k$, we get pure quantifier elimination. For $s_i = n_i$ for all $i = 1, \ldots, k$, we get closer to the CAD cells (especially if we chose Ψ_3).

Example 3.8. *We consider the virtual substitution tree from Example 3.1.*

Let us consider the subtree for the test candidate x_3 in the tree from Figure 3.2. If we eliminate x, we would get the partial explanation (using Ψ_1)

$$2 + \frac{\sqrt{2}}{2} < x \to \mathit{false}$$

which excludes the dashed region as shown in Figure 3.15

Compare this to the result that we obtain when stopping after eliminating y:

$$\mathit{true} \to 2(x-2)^2 < 1$$

which excludes the dashed region as shown in Figure 3.16 being much more general as it excludes not only the region enclosing the current assignment but also other regions.

As mentioned, choosing Ψ_3, eliminating all variables and also fixing the order of zeros of φ in y, we obtain a CAD cell.

$$x = y \wedge 2 + \frac{\sqrt{2}}{2} < x \to \mathit{false}$$

which excludes the dashed region as shown in Figure 3.17. Note that without adding an interval description for y, we get cylindrically ordered regions.

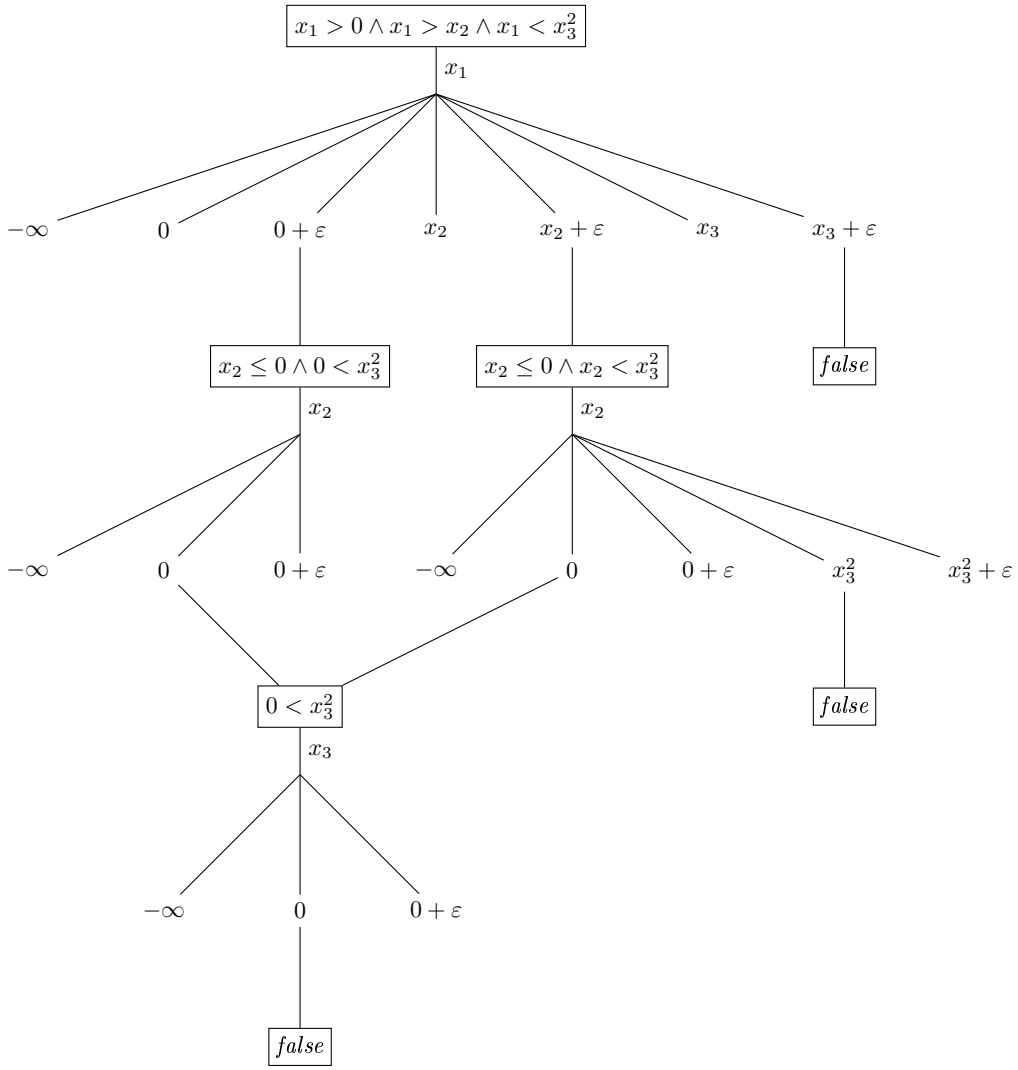

Figure 3.12: Virtual substitution tree: ambiguous representatives behave the same

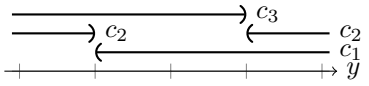

Figure 3.13: Solution intervals for each constraint in y

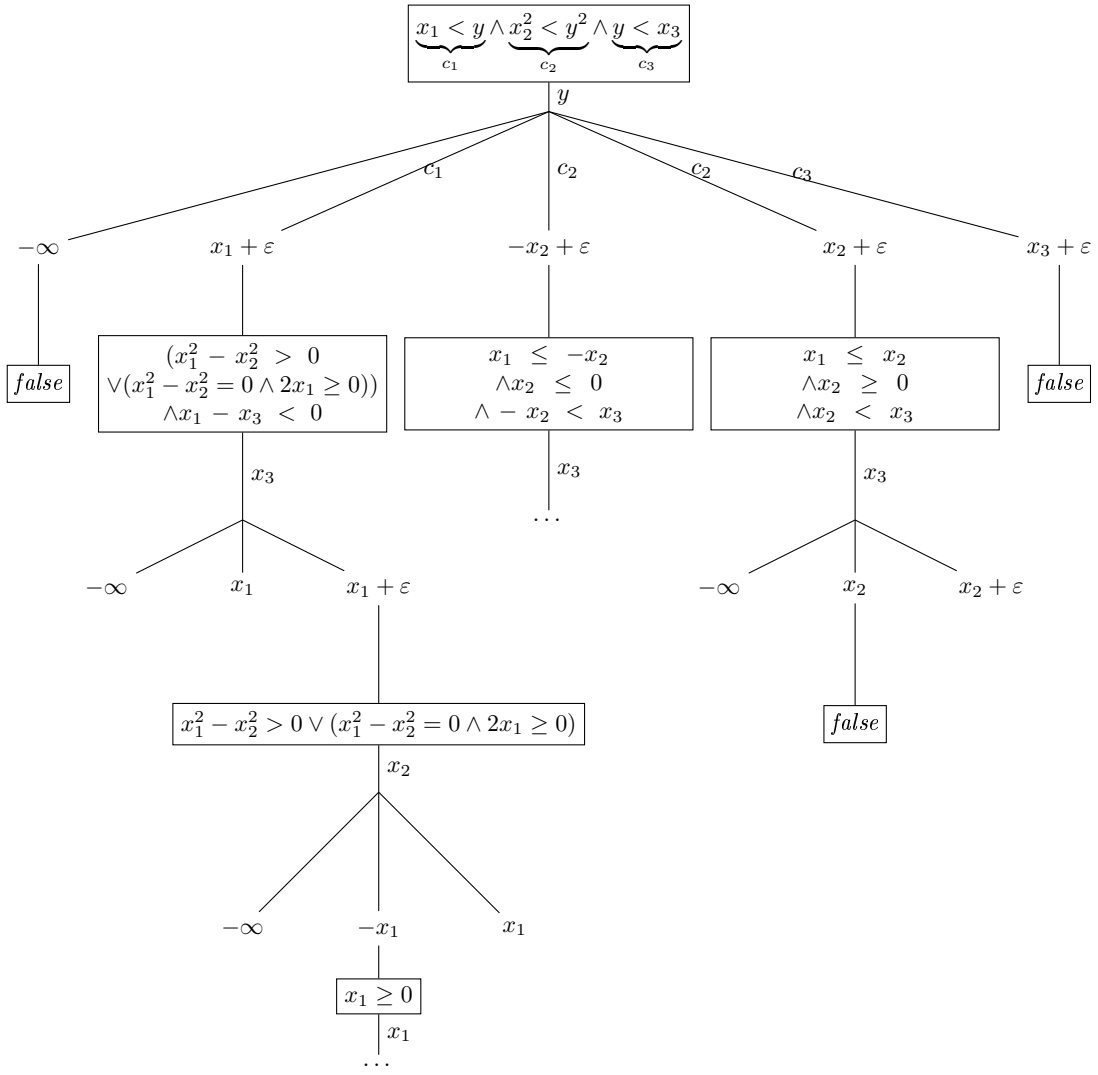

Figure 3.14: Partial virtual substitution tree: Nodes containing *false* can be obtained before eliminating all variables

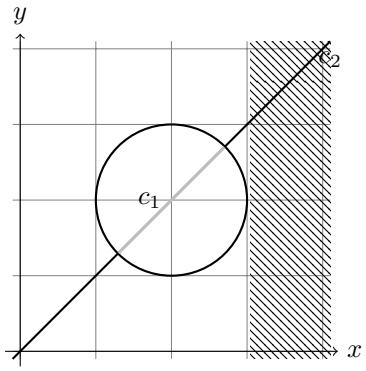

Figure 3.15: Region excluded when eliminating all variables

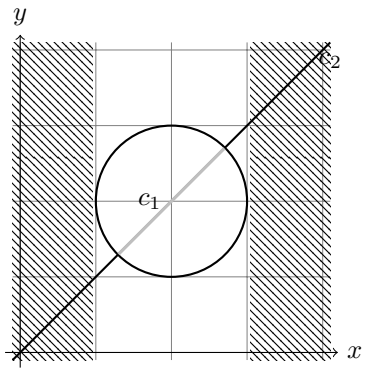

Figure 3.16: Region excluded when stopping at the first node

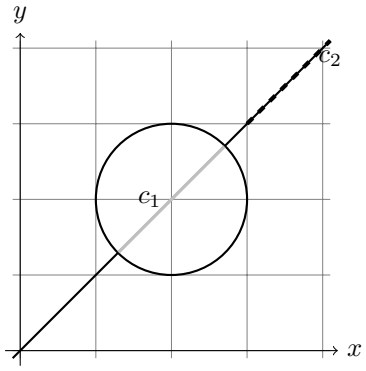

Figure 3.17: Defining all variable intervals with Ψ_3 results in a CAD cell

Chapter 4

Comparison

To show the benefits of our method, we compare it with the cylindrical algebraic decomposition for mcSAT as described in [JdM12]. We examine the methods by the formula given in Example 2.1.

For our example, we denote the state of the mcSAT solver as a sequence $\langle \dots \rangle$ of constraints (of the form $p \sim 0$) and variable assignments (of the form $x \mapsto \alpha(x)$). We assume that all literals contained in the sequence are consistent with the assignments in the trail (i.e. they do not evaluate to *false*). If we find that there exists no assignment for a variable y without violating this property, we denote a call to the explanation function with $\langle \dots \rangle \vdash \mathit{explain}(D,\alpha)$ where D is a minimal subset of the sequence's constraints causing the conflict.

Furthermore, the CAD procedure uses the notation $\mathit{zero}_x(i,p)$ to represent the ith zero of the polynomial p with possibly polynomial coefficients in the variable x.

We first examine how the problem is solved using the CAD:

Example 4.1 (CAD for mcSAT)**.** *Let the input formula be defined as:*

$$\varphi \equiv \underbrace{(x-2)^2 + (y-2)^2 - 1 < 0}_{c_1} \wedge \underbrace{x - y = 0}_{c_2}$$

We initialize the solver and try to find satisfying variable assignments:

$$\begin{aligned} \langle c_1, c_2 \rangle &\xrightarrow{\textit{assign } x} \langle c_1, c_2, x \mapsto 0.5 \rangle \\ &\xrightarrow{\textit{no ext. for } y \textit{ exists}} \langle c_1, c_2, x \mapsto 0.5 \rangle \vdash \mathit{explain}(c_1, \alpha) \end{aligned}$$

We obtain the explanation lemma

$$c_1 \to \neg(x < \mathit{zero}(1, x^2 - 4x + 3))$$

resulting in the dashed region as in Figure 4.1.

Thus, the explanation is conflicting with the current assignment and we have to backtrack before adding the description of the excluded cell to the sequence.

$$\begin{aligned} \dots &\xrightarrow{\textit{backtrack}} \langle c_1, c_2, c_3 : \neg(x < \mathit{zero}_x(1, x^2 - 4x + 3)) \rangle \\ &\xrightarrow{\textit{assign } x} \langle c_1, c_2, c_3 : \neg(x < \mathit{zero}_x(1, x^2 - 4x + 3)), x \mapsto 1.2 \rangle \\ &\xrightarrow{\textit{no ext. for } y \textit{ exists}} \langle c_1, c_2, c_3, x \mapsto 1.2 \rangle \vdash \mathit{explain}(c_1 \wedge c_2, \alpha) \end{aligned}$$

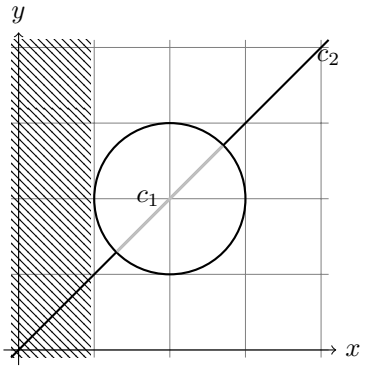

Figure 4.1: Excluded regions

We obtain the explanation

$$c_1 \wedge c_2 \rightarrow \neg(zero_x(1, x^2 - 4x + 3) < x < zero_x(1{,}2x^2 - 8x + 7))$$

adding the dashed region as shown in Figure 4.2.

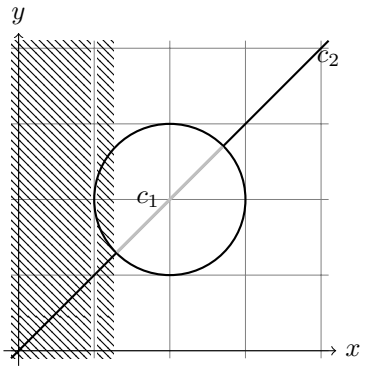

Figure 4.2: Excluded regions

And continue search:

$$\begin{aligned}
\ldots &\xrightarrow{backtrack} \langle c_1, c_2, c_3, c_4 : \neg(zero_x(1, x^2 - 4x + 3) < x < zero_x(1{,}2x^2 - 8x + 7))\rangle \\
&\xrightarrow{assign\ x} \langle c_1, c_2, c_3, c_4, x \mapsto 2\rangle \\
&\xrightarrow{assign\ y} \langle c_1, c_2, c_3, c_4, x \mapsto 2, y \mapsto 2\rangle \\
&\xrightarrow{assignment\ complete} SAT
\end{aligned}$$

In the following we see how our method can be modified to emulate the CAD.

Example 4.2 (VS for mcSAT). *We generate the first explanation from the previous example explain(c_1,α) using our procedure as shown in Figure 4.3:*

$$c_1 \to \neg(x < 1)$$

resulting in the same excluded region as the CAD shown in Figure 4.1.

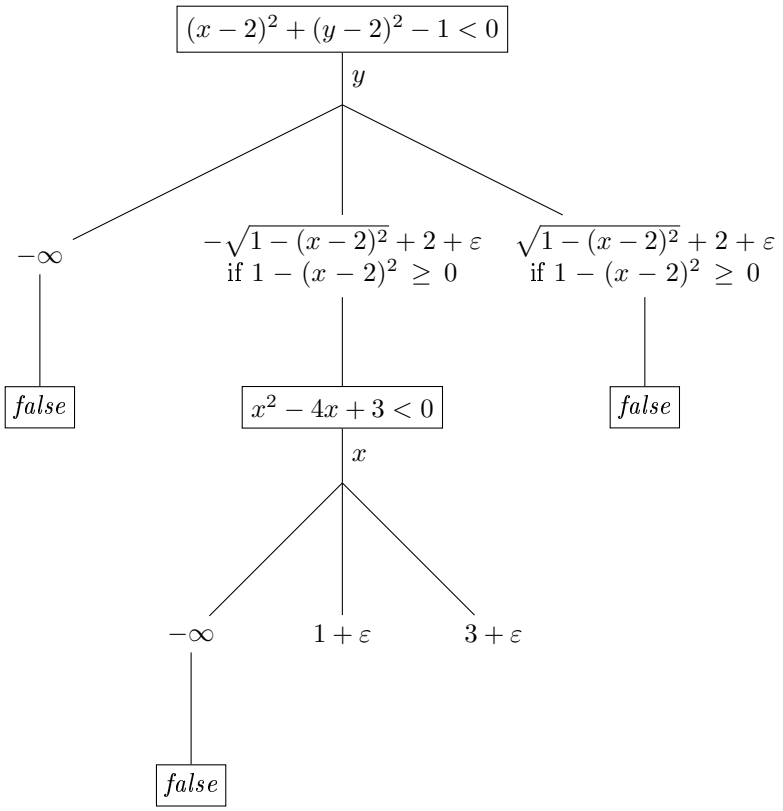

Figure 4.3: Partial virtual substitution tree

The second explanation explain($\{c_1,c_2\}$,α) can be generated using the tree in Figure 2.2. Again, we obtain a similar explanation.

$$c_1 \wedge c_2 \to \neg(x < 2 - \frac{\sqrt{2}}{2})$$

But because the order of the zeros of φ in y has not been fixed, we obtain a larger region as shown dashed in Figure 4.4.

So far, we generated complete paths, being specific to the current conflict. In the following we see more general explanations generated by the pure quantifier elimination approach:

Example 4.3 (VS for mcSAT). *If we would abort the procedure shown in Figure 4.3 after eliminating y, we would obtain the explanation*

$$c_1 \to x^2 - 4x + 3 < 0$$

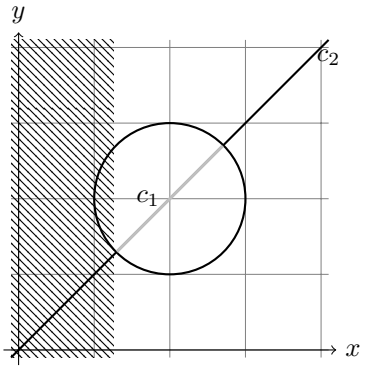

Figure 4.4: Excluded regions

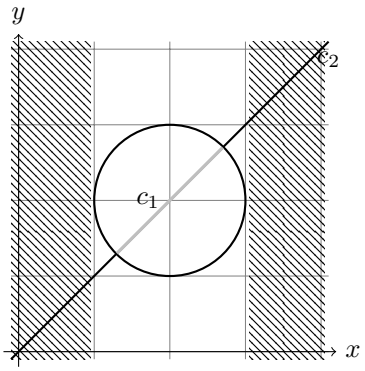

Figure 4.5: Excluded regions

resulting in the dashed region as in Figure 4.5.

We do the same for the second explanation explain($\{c_1,c_2\},\alpha$) *using the tree in Figure 2.2:*

$$c_1 \wedge c_2 \rightarrow 2x^2 - 8x + 7 < 0$$

excluding the dashed region as shown in Figure 4.6.

We saw, that when eliminating all variables, we obtain explanations similar to the CAD. But even then, we have the chance to get a better explanation, because first, virtual substitution does not require the decomposition to be cylindrical but only sign-invariant and second, as observed in Section 3.2.6, it is possible that virtual substitution can detect unsatisfiability earlier and variables that do not cause the conflict for a subtree may be skipped. Aside from that, the CAD is more expensive than the VS.

Furthermore, the last example shows us, that stopping our procedure earlier may exclude even bigger regions. This is possible, because virtual substitution keeps the sign of the polynomials due to its symbolic computations. Those explanations may be computational harder to use for theory decision as those where all variables have been eliminated, but they are more general. Moreover, as seen in Section 3.2.7, we can vary between these two extremes as we can stop our procedure at any time. This flexibility allows us to decide by a heuristic where it is best to stop.

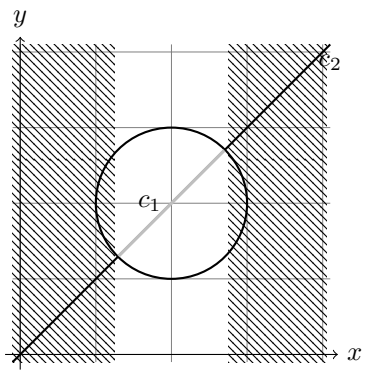

Figure 4.6: Excluded regions

Chapter 5

Future work

5.1 Enforcing conjunctions in nodes

The current approach allows arbitrary formulas in the nodes of the virtual substitution tree. Although the input of the explanation function is given as a conjunction of constraints, disjunctions are introduced by the substitution rules, because each of them consists of a case distinction. One can observe for each rule, that given a full assignment, at most one of these cases evaluates to *true*.

It is tempting to enforce conjunctions in each node because this has an advantage: Theorem 3.9 says that substituting representatives that are not in the set of test candidates into the formula obtains *false*. This would potentially lead to smaller partial virtual substitution trees and also, since we know the substitution result for those representatives, we can save one substitution step. However, note that the regions still have to be defined using Ψ_1, Ψ_2 or Ψ_3 as the order or existence of the zeros is still not implied as stated in Example 3.2.

One idea is to add a new node for each case introduced by a substitution rule and handle each subtree separately. This would lead to less constraints in one node and therefore less substitutions. It is still to be figured out how a formula can be generated and which parts of the tree can be left out.

5.2 Determining constraints relevant to a conflict

It is already pointed out in Section 3.2.6 that some variables are not part of the reason for a conflict in the subtree for all choices of the test candidate for the unassigned variable. Or more precise, not all constraints are part of the reason for a conflict in such a subtree.

We can determine those constraints by evaluating each constraint c_j occurring in the input formula $\varphi \equiv c_1 \wedge \cdots \wedge c_m$ under the choice of a test candidate $t \in tcs(y,\varphi)$ for the unassigned variable y and the current assignment α:

$$[\![c_j[t//y]]\!]^\alpha$$

If the constraint does not evaluate to *false*, it is not part of the conflict in the subtree corresponding to $\varphi[t//y]$.

The basic idea is now to take only one of these constraints for further subtree generation, which is justified as follows:

$$\varphi[t//y] \equiv \bigwedge_{j=1,\ldots,m} c_j[t//y] \models c_k[t//y]$$

for each $k = 1, \ldots, m$.

Example 5.1. *Consider Example 3.7 again. Applying our idea to* $\varphi[x_1 + \varepsilon//y]$ *results in* $(x_1^2 - x_2^2 > 0 \vee (x_1^2 - x_2^2 = 0 \wedge 2x_1 \geq 0))$*, application to* $\varphi[-x_2 + \varepsilon//y]$ *results in* $x_2 \leq 0$ *and application to* $\varphi[x_2 + \varepsilon//y]$ *results in* $x_2 < x_3$.

This approach would not only reduce the number of variables in each subtree, but also reduce the number of possible representatives in each elimination step resulting in smaller path descriptions and bigger regions that can be excluded.

However, it is still to be figured out if and how this observation can be combined with our procedure regarding correctness and completeness.

5.3 Allowing square root expressions as input

It is desirable to obtain a much more flexible and universal procedure by lifting the restrictions to the solver mentioned in Section 3.2.1. One restriction on the solver is that we do not allow square root expressions as input.

Input constraints can be transformed to an input formula without square root expressions as follows: Let c be a constraint containing a square root expression t. Then let c' be the constraint obtained by replacing each occurrence of t in c by the variable x. Then

$$c'[t//x] \equiv c$$

is a formula that does not contain t. Applying the same procedure on each constraint and square root expression results in an equivalent formula without square root expressions.

What is to be proven is

- that this procedure is possible for each constraint containing (multiple) square root expressions (i.e. virtual substitution rules are applicable for *all* square root expressions in a constraint) and
- that we still obtain a finite basis.

5.4 Allowing constraints obtained by different elimination orders as input

Without any restrictions to the mcSAT solver, the assignment order of the variables can change and thus, constraints generated using different elimination orders (and different to the current elimination order) may be passed as input to the explanation function. Thus, we cannot assume a fixed elimination order and have to alter the definition of an elimination step in the proof of Theorem 3.8:

- Let $l_1, l_2 \in S$ with $x \in \mathit{Vars}(l_1) \cap \mathit{Vars}(l_2)$ occurring at most quadratically in l_1 and l_2 and $t \in \mathit{rs}(x, l_1)$ (where x, l_1, l_2, t have not been chosen together before)
- $s := l_2[t//x_i] \wedge \mathit{sc}(t)$
- $\mathit{Subst}(S) := \mathit{Constraints}(s)$

This has a crucial consequence: The bound given in Equation 3.4 is not valid anymore because we do not have a fixed elimination order anymore. Thus constraints where a variable x has been eliminated can produce a new constraint containing x when doing a substituting step with another constraint containing x.

Moreover, we can give a counterexample, i.e. the closure of virtual substitution is infinite:

Example 5.2.

- $C_0 := \{\underbrace{x_1 = 2}_{c_1}, \underbrace{x_1 = 2x_2}_{c_2}, \underbrace{x_2 = 2x_1}_{c_3}\}$
- *Substituting* $x_1 = 2$ *from* c_1 *into* c_2 *gives* $c_4 : x_2 = 1$
- *Substituting* $x_2 = 2x_1$ *from* c_3 *into* c_4 *gives* $c_5 : 2_x1 = 1$
- *Substituting* c_2 *into* c_5 *gives* $c_6 : 4x_1 = 1$
- *Substituting* c_3 *into* c_6 *gives* $c_7 : 8x_2 = 1$
- ...

The intuitive cause for the infinite set of constraints in the example above is that those constraints are unsatisfiable together. If they would be eliminated altogether in a conjunction, we would get *false* as result not later than after a few steps.

In our setting, the input is a conjunction of constraints, and we can assume a minimal set of conflicting constraints. Perhaps, we do not need to rely on the closure of virtual substitution and get the desired statement somehow else. It is still open if any restrictions (weaker than assuming a fixed variable order) to the constraints used in a substitution step can be made to obtain a finite basis.

5.5 Combination of the VS and the CAD for generating explanations

We think that the most efficient and effective explanation methods can be obtained by combining multiple decision procedures. This way each procedure can be applied to (sub)problems where it performs best. Here, our goal is it to combine the CAD with the VS, by applying the VS on problems where a variable occurs at most quadratically in the input formula and switch to the CAD whenever the VS becomes infeasible. This way, we would get a complete method with the benefits of the VS for those cases where it is applicable.

One issue that is to be solved is the different representation of zeros of a polynomial in a variable: While our procedure makes use of square root expressions, the CAD for mcSAT as described in [JdM12] employs the following notation:

Definition 5.1 (Zero expression)**.** *Let p be a (possibly multivariate) polynomial containing a variable x, then the* ith zero of p in x *is denoted as* $zero_x(i,p)$.

The CAD adaption always eliminates all variables and defines cylindrically ordered regions. As this adaption also assumes a fixed order on the variables in which they are assigned, the assignment passed to the explanation function always implies the same ordering of the zeros in a given variable, and thus $zero_x(i,p)$ refers always to the same zero of p in x.

In contrast, the VS embedding cannot make this assumption: Even with a fixed order of variable assignments, in general only the symbolic representation (given as a square root expression) of a zero is known. It is possible that two assignments given as input to the procedure imply different

orderings of the zeros in a variable, because the decomposition induced by Ψ_1, Ψ_2 and Ψ_3 is not cylindrical, and at the latest because not all variables are eliminated in each case.

Because of the different semantics of the zero representations, we cannot unify them directly. However, we could convert them to the appropriate representation if they occur in an input to *explain*:

- If a square root expression t occurs in the input to the CAD adaption, we can order all zeros of the originating polynomial p in the variable x under the current assignment and determine the position i of t in the resulting ordering. Thus we can use $zero_x(i,p)$ as representation.
- If a $zero_x(i,p)$ occurs in the input to the VS embedding, we can order all zeros of p and take the symbolic description (given as a square root expression t) of the ith zero according to the resulting ordering.

However, we do not know if this approach is correct.

An alternative approach to that would be to transform constraints containing zero expressions to equisatisfiable formulas that are part of the logic:

- Square root expressions could be transformed as described in Section 5.3 when generating the explanation.
- If an expression of the form $zero_x(i,p)$ occurs in the input to the virtual substitution, its semantics could be encoded using a non-linear arithmetic formula by making a case distinction between $i = 1$ and $i = 2$ and between the symbolic zeros of constant, linear and quadratic polynomials respecting their side conditions.

Chapter 6

Conclusion

As the CAD implementation for mcSAT already showed promising results in the Z3 solver [JdM12] for solving non-linear arithmetic, we think that our embedding of the virtual substitution method can improve these results for cases where the virtual substitution is applicable, based on our observations in Section 4.

Once this embedding is implemented, the different variants of our method for generating explanations can be compared: First, the depth of the generated virtual substitution tree, and second, the formula used to define the path in the generated tree representing the current assignment. Using these results, we can develop and evaluate ideas for designing a heuristic regarding their effectiveness and efficiency.

The CAD adaption for mcSAT is currently being implemented as part of the SMT-RAT [Cor16] solver. By implementing also the VS embedding as a module for SMT-RAT, the theory modules, e.g. the ones responsible for assigning theory values to variables, can be shared between the CAD and VS implementation. This common base will also allow the combination of the two decision procedures to obtain a complete explanation function with advantages from the virtual substitution.

Bibliography

[CA11] Florian Corzilius and Erika Abraham. Virtual substitution for smt solving. In *18th Int. Symp. on Fundamentals of Computation Theory (FCT'11)*, volume 6914 of *LNCS*, pages 360–371. Springer Berlin Heidelberg, 2011.

[Col75] G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Automata Theory and Formal Languages*, volume 33 of *LNCS*, pages 134–183. Springer, 1975.

[Cor16] Florian Corzilius. *Integrating Virtual Substitution into Strategic SMT Solving.* PhD thesis, RWTH Aachen University, 2016.

[DLL62] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962.

[DMJ13] Leonardo De Moura and Dejan Jovanović. A model-constructing satisfiability calculus. In *International Workshop on Verification, Model Checking, and Abstract Interpretation*, pages 1–12. Springer, 2013.

[DP60] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960.

[GGI+10] S. Gao, M. K. Ganai, F. Ivancic, A. Gupta, S. Sankaranarayanan, and E. M. Clarke. Integrating ICP and LRA solvers for deciding nonlinear real arithmetic problems. In *Proc. of FMCAD'10*, pages 81–89. IEEE, 2010.

[HR97] S. Herbort and D. Ratz. Improving the efficiency of a nonlinear-system-solver using a componentwise Newton method. Technical Report 2/1997, Inst. für Angewandte Mathematik, University of Karlsruhe, 1997.

[JdM12] D. Jovanovic and L. M. de Moura. Solving non-linear arithmetic. In *Proc. of IJCAR'12*, volume 7364 of *LNAI*, pages 339–354. Springer, 2012.

[Wei97] V. Weispfenning. Quantifier elimination for real algebra — the quadratic case and beyond. *Applicable Algebra in Engineering, Communication and Computing*, 8(2):85–101, 1997.